2016-2324

IN THE
UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT

REMBRANDT PATENT INNOVATIONS, LLC AND REMBRANDT SECURE COMPUTING, LP,

Plaintiffs-Appellants,

v.

APPLE INC.,

Defendant-Appellee.

Appeals from the United States District Court for the Northern District of California in Case Nos. 3:14-cv-05093-WHA, 3:14-cv-05094-WHA
Judge William H. Alsup

Corrected Brief of Plaintiffs-Appellants Rembrandt Patent Innovations, LLC and Rembrandt Secure Computing, LP

Jacob A. Schroeder
Finnegan, Henderson, Farabow,
Garrett & Dunner, LLP
3300 Hillview Avenue
Palo Alto, CA 94304-1203
(650) 849-6600

E. Robert Yoches
Finnegan, Henderson, Farabow,
Garrett & Dunner, LLP
901 New York Avenue, NW
Washington, DC 20001-4413
(202) 408-4000

Attorneys for Plaintiffs-Appellants
Rembrandt Patent Innovations, LLC
and Rembrandt Secure Computing, LP

September 26, 2016

Claim Language at Issue

1. An architecture for initializing a computer system comprising:

a processor;

an expansion bus coupled to said processor;

a memory coupled to said expansion bus, said memory storing a system BIOS for execution by said processor upon power up of the computer system;

a plurality of boot components coupled to said expansion bus and accessed by said processor when said system BIOS is executed;

a trusted repository coupled to said expansion bus; and

means for verifying the integrity of said boot components and said system BIOS wherein integrity failures are recovered through said trusted repository.

4. A method for initializing a computer system comprising the steps of:

(1) invoking a Power on Self Test (POST);

(2) verifying the integrity of a system BIOS;

(3) verifying the integrity of a boot component; and

(4) when said boot component fails, recovering said failed boot component.

Appx72-73.

UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT
Appeal No. 16-2324

Rembrandt Patent Innovations v. Apple, Inc.

Certificate of Interest

Counsel for the appellants, Rembrandt Patent Innovations, LLC and Rembrandt Secure Computing, LP, certifies the following (use "None" if applicable; use extra sheets if necessary):

1. The full name of every party represented by me is:

   Rembrandt Patent Innovations, LLC
   Rembrandt Secure Computing, LP

2. The name of the real party in interest (Please only include any real party in interest NOT identified in Question 3) represented by me is:

   None

3. All parent corporations and any publicly held companies that own 10 percent or more of the stock of the party represented by me are:

   Rembrandt II, LLLP
   Rembrandt IP Management, LLC

4. The names of all law firms and the partners or associates that appeared for the party or amicus now represented by me in the trial court or are expected to appear in this court (and who have not or will not enter an appearance in this case) are:

   Finnegan, Henderson Farabow, Garrett & Dunner, LLP.
   Anita Bhushan, Christopher Blackford, Alyssa Holtslander, Gerald Ivey, Stephen Kabakoff, Robert McCauley, Richard Racine, Benjamin Schlesinger, Jeff Watson

   Yarbrough Wilcox, PLLC
   Herbert ("Trey") Yarbrough

Table of Contents

Table of Authorities

Page(s)

Cases

Statutes

Statement of Related Cases

No appeal in or from the same civil action was previously before this or any other appellate court. Counsel are not aware of any case that may be directly affected by this Court's decision.

Statement of Jurisdiction

The statutory basis for the district court's jurisdiction was 28 U.S.C. § 1338(a). The statutory basis for this Court's jurisdiction to hear the issues in this appeal is 28 U.S.C. § 1295(a)(1). The district court issued a final judgment on June 10, 2016, and Rembrandt filed a notice of appeal on July 7, 2016.

I. Statement of the Issues

1. Whether the district court improperly limited the term "recover[y]" in claims 1 and 4 of United States Patent No. 6,185,678 to "automatic recovery" "without human intervention" when nothing in the claims requires such a limitation; the patent is about the need to preserve security, not automation; the specification and the prosecution history not only lack any clear and unmistakable disclaimer of all human intervention, but the specification describes recovery using human intervention; and the USPTO considered recovery to include human intervention?

2. Whether the district court erred by granting summary judgment of noninfringement where the undisputed evidence showed that the accused products practiced recovery under the district court's construction, both literally and by equivalents, and at a minimum there are genuine disputes of material fact on this issue, including unchallenged opinions of Rembrandt's expert based on statements by Apple's witnesses, source code, and documents?

II. Preliminary Statement

The ’678 Patent describes the result of research into computer security at the University of Pennsylvania in the mid-1990s. That research resulted in a computer architecture that checks the integrity of each layer of startup software before executing it, except for a small portion that is “trusted” because it is in a read-only memory that cannot be modified. After the computer executes that small portion of trusted software, it begins loading and executing the next layers (the “boot process”), verifying the integrity of each layer before execution, and thereby creating a chain of trust. If a layer fails its verification, the ’678 Patent describes various ways to recover without making the computer system vulnerable to attack.

The district court waited to construe claim terms until just before trial when it decided Apple’s motion for summary judgment of noninfringement, at which time the court limited “recover[y]” in the claims to “automatic recovery” with no human intervention. The court committed two errors that likely arose because it focused only on the term “recovery,” rather than the intrinsic evidence as a whole, and this focus appears to have led the court to misunderstand the invention as involving automation, not security.

The first error involved the court’s limitation of the claim term “recovery.” Although no claim defines how to recover failed components, the court limited each claim to “automatic” recovery foreclosing any human intervention, even

plugging in the device. Appx10. In doing so, the court mistook the purpose of the invention as automation, rather than security.

The court's Order identified nothing in the specification amounting to a clear and unmistakable disclaimer, holding instead that the inventors "implicit[ly]" disclaimed all nonautomatic recovery processes based on its erroneous conclusion that the specification described no instances of human intervention in recovery. Appx12-13. The court improperly dismissed several discussions in the specification of human intervention during recovery and ignored the prosecution history showing that both the USPTO and the inventors understood the claims to include human intervention.

The second error involves the district court's improper grant of summary judgment that ignored unrebutted evidence that Apple's accused products performed recovery even under the court's construction. Rembrandt's expert, Dr. Tygar, provided detailed and unrebutted opinions that Apple's accused products perform automatic recovery consistent with the district court's interpretation, and he based his opinion on (1) testimony from Apple's corporate representative about the operation of the accused devices, (2) testimony from other Apple witnesses that a user plays no role during the recovery of the accused devices, (3) Apple's source code showing no user input during recovery, and (4) Apple's documents describing the security features with no user intervention. The

district court did not even acknowledge Dr. Tygar's opinions or the evidence he relied on, nor did it explain why that evidence at least failed to present a genuine issue of material fact. The district court also failed to address unrebutted evidence of infringement by equivalents for "automatic" recovery.

This Court should reverse the district court's holding that the inventors "implicit[ly]" disclaimed recovery processes involving human activity, and give the claim terms "recovered" (claim 1) and "recovering" (claim 4) their plain and ordinary meaning. The Court should also vacate the district court's grant of summary judgment that flowed from this holding. At the very least, given the issues of material fact, this Court should vacate the judgment of noninfringement and remand this case for a trial on whether Apple's accused devices infringe literally or under the doctrine of equivalents.

III. Statement of the Case

The '678 Patent, "Secure and Reliable Bootstrap Architecture," describes an invention Drs. Arbaugh, Smith, Keromytis, and Farber made in computer security at the University of Pennsylvania (Penn) in the mid-1990s. Appx1607; Appx1644. Arbaugh, who worked for the National Security Agency as a Senior Computer Scientist (*see* Appx1607), was in a Ph.D. program at Penn in Computer Science (*id.*). His faculty advisers were Drs. Smith and Farber, who were also Keromytis's

advisers. Appx1644. The invention in the patent came from their collective work, some of which was for Arbaugh's Ph.D. thesis. Appx1607.

A. Technological Background: Unsecure Bootstrap Processes Jeopardize Computer Integrity.

One of the first actions a computer performs after being turned on is to load a software component called a "boot loader," which then loads other software components. At the time of the invention, a computer was vulnerable to unauthorized or malicious software during this "boot" process because the computer was not yet running security software, such as antivirus software, that protects against intrusions.

Figure 1a (below) from the '678 Patent shows a typical boot process at the time of the invention. The architecture included several "boot components": system BIOS[1] 112, expansion ROMs[2] 122, boot block[3] 132, and operating system kernel[4] 142. *See, e.g.*, Appx62(1:53-58). Figure 1a organizes these boot components as functional layers 110, 120, 130, and 140, which each include

[1] "BIOS" stands for "basic input/output system," which is software that performs startup operations.

[2] ROM stands for "read-only memory."

[3] A "boot block" is software that performs part of the boot process during initialization.

[4] The operating system is software that manages computer hardware and software resources, and provides services for computer programs. The "kernel" is a basic portion of the operating system.

software to perform functions before loading the next layer. *See* Appx50(Fig.1a); Appx62(1:27-44); Appx65(7:56-8:37).

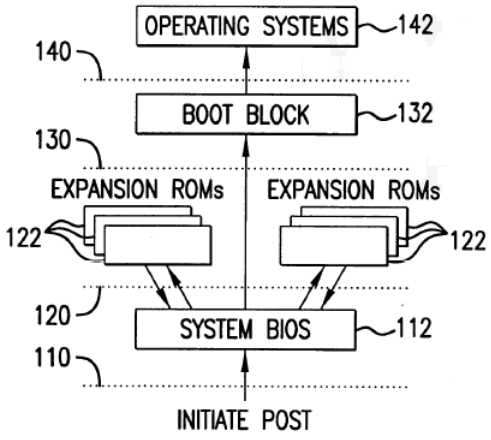


FIG. 1a

Appx50(Fig.1a).

First layer 110 includes system BIOS 112. Appx65(7:61-62). System BIOS 112 passes control to software in ROMs 122 in second layer 120, which returns control to BIOS 112. Appx65(8:12-22). BIOS 112 then passes control to third layer 130, which contains boot block 132. Appx65(8:23-29). Boot block 132 passes control to fourth layer 140 containing the operating system kernel 142 (Appx65(8:29-31)), which then loads the remainder of the operating system.

Software is "trusted" when one can ensure the absence of any unauthorized modification, such as by malicious software. An operating system lacks trust if untrusted software invokes it, because that operating system may invoke or permit execution of malicious or infected software. *See, e.g.*, Appx62(1:39-52).

The prior art recognized the need to secure the operating system once it was running, such as by using antivirus software, but paid inadequate attention to securing the bootstrap process. *See* Appx62(1:44-52); Appx63(3:4-25). The ’678 Patent describes some attempts to provide a secure boot environment, but these early attempts either failed to verify the integrity of all bootstrap components sufficiently, or undermined system integrity by permitting installation of unverified bootstrap components. Appx62-63(2:1-3:25). Other attempts proposed impractical architectural revisions of standard PC architecture (*see, e.g.,* Appx62(2:6-9)), or failed to provide an adequate recovery process, especially for bootstrap components (Appx63(3:40-42)). [5]

One recovery effort required someone to find and insert a floppy disk with a replacement component (Appx63(3:42-45)), which presented security problems unrelated to automation. “[P]roviding physical security for the floppy disk is extremely difficult [because] [u]sers can take the disks wherever they like, and do whatever they like to them.” Appx63(3:47-50). This method also “only focused on repairing a single component of the entire process, i.e. only repairing the boot block, or the BIOS but not both.” Appx63(3:54-57). The ’678 Patent contrasted this approach of only repairing a single component “to the present invention which

[5] Recovery refers to the process of replacing a failed component with a replacement to keep the boot process secure.

provides automatic recovery of *all* of the bootstrap components including ROM chips." Appx63(3:57-59) (emphasis added).

B. The '678 Patent: A Secure and Reliable Bootstrap Architecture Solves Problems with Prior-Art Bootstrap Processes.

1. The specification describes a system that verifies integrity on a layer-by-layer basis.

The '678 Patent describes an architecture where transitions between software boot layers occur only after verifying the next layer's integrity. *See, e.g.*, Appx66(9:35-39). The Abstract explains the "basic principle" of the present invention as "sequencing the bootstrap process as a chain of progressively higher levels of abstraction, and requiring each layer to check a digital signature of the next layer before control is passed to it." Appx48.

The district court described this aspect of the claimed invention as follows:

> The '678 patent is entitled "Secure and Reliable Bootstrap Architecture" and describes techniques for booting a computer system in a secure manner. The patented invention involves a "chain of integrity checks" beginning with the foundational layer of the computer system—certain important hardware and the Basic Input/Output System ("BIOS")—which is assumed to be secure. Relying on that assumption, once the computer is powered on, the BIOS verifies the integrity (that is, confirms it is safe to load) of the next layer in the boot sequence before passing control of the system to that layer (which repeats the process for the subsequent layer), a process referred to as "bootstrapping."

Appx3.

Figure 2a (below) shows one embodiment of the boot process.

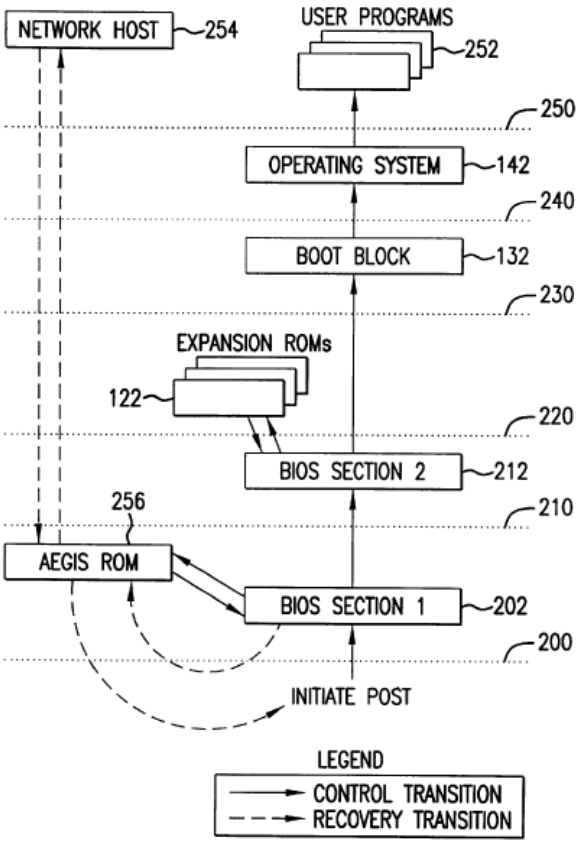


FIG. 2a

Appx53(Fig.2a).

"A major design decision is the consequence of a failed integrity check." Appx48. At one end of the spectrum, the system could "simply halt the bootstrap process." *Id.* At the other end, "the bootstrap process of the present invention can be augmented with automated recovery procedures which preserve the security properties of the bootstrap process of the present invention under the additional assumption of the availability of a trusted repository." *Id.* "A variety of means by

which such a repository can be implemented are disclosed with attention focused on a network-accessible repository." *Id.*

The Summary of the Invention describes the focus of the invention as preserving the integrity of the computer's bootstrap process. Appx63(4:33-35). "Integrity is validated at each layer transition in the bootstrap process and a recovery process is included for integrity check failures." Appx63(4:35-37).

If the integrity of the next layer cannot be verified, the patent describes various ways of recovery that replace the failed component, including the use of a trusted repository. The district court observed that the patent explains "storing replacements for the various components of the boot sequence in a trusted location [for use] to repair any integrity failures." Appx3.

One of the embodiments, AEGIS, which served as Dr. Arbaugh's proof of concept, "construct[s] a chain of integrity checks, beginning at power-on and continuing until the final transfer of control from the bootstrap components to the operating system itself." Appx64(6:6-13).

AEGIS also includes a recovery mechanism. "In the AEGIS boot process, either the operating system kernel is started, or a recovery process is entered to repair any integrity failure detected. Once the repair is completed, the system is restarted (warm boot) to ensure that the system boots." Appx66(10:4-8). The "warm boot" requires pressing the ctrl-alt-del keys, which involves user

intervention. Appx65(8:2-5). The patent continues that "[t]his entire process occurs without user intervention" Appx66(10:8), showing that the phrase "without user intervention" permits some user action in recovery, such as pressing the ctrl-alt-del keys for a "warm boot."

2. The specification provides a range of options for recovery.

The Summary discloses modifying the recovery process to fit the needs of a particular situation. The recovery process can use "a secure protocol to inform a trusted repository that a failure has occurred and to obtain a valid replacement component." Appx63(4:48-51). "[I]f security is not a concern, then a less robust approach could be used." Appx63(4:56-59). The Abstract describes the implementation of a recovery process as a "design decision." Appx48.

One option is to place the replacement software in read-only memory to prevent tampering. Appx66(10:44-61). After replacing the failed component, the system restarts with the layered integrity checks. Appx66(10:63-67). The Summary makes clear, however, that this type of automatic recovery is an option. "The present invention *can also be utilized* to reduce the Total Cost of Ownership (TCO) of a personal computer, through automatically detecting and repairing integrity failures, thereby permitting the user to continue to work without the nuisance of a trouble call to support staff and the associated down time." Appx63(4:60-65) (emphasis added). Although the specification allows human participation during

recovery, it does not allow a person to compromise security in the recovery process.

The ’678 Patent never equates “automatically detecting and repairing integrity failures” with an absence of all human involvement. Instead, it explains how automatic-recovery options may include human involvement in recovery, such as by creating a log to identify computers requiring “hands on” repairs so a system administrator can later schedule recovery. Appx63-64(4:65-5:3).

Even in AEGIS, besides the “warm boot” in recovery, there is human intervention because the user must choose how to proceed when the trusted repository is unavailable.

> In each case, AEGIS attempts to recover from a trusted repository, step 298, as discussed below. Should a trusted repository be unavailable after several attempts, then the client’s further action depends on the security policy of the user. For instance, a user may choose to continue operation in a limited manner or may choose to halt operations altogether.

Appx66(10:19-25).

When explaining one of the passages referring to “[a]utomatically detecting and repairing integrity failures,” the specification acknowledges recovery may require “hands on” repairs to certain boot components such as ROMs, and scheduling those repairs when the user is not using the computer. Appx71(20:45-54). Recovery is not complete until the hands-on repair is finished.

3. Neither the USPTO nor the patentee treated "recovery" as precluding human interaction.

During prosecution, the examiner rejected claims 1-7 as obvious in view of, *inter alia*, U.S. Patent No. 5,564,054 ("Bramnick"). Appx1469-1470. The examiner explained, "Bramnick teaches that it is known to provide a trusted repository (boot registry) and recover through the trusted repository as set forth with figures 3A-3B, at column 3, lines 24-63 and at column 7, lines 18-35." Appx1470. The cited portion of Bramnick reads:

> Assuming the minimal system hardware and software components are present and undamaged, the computer will boot using this minimal configuration and the user can then edit the boot registry that interrupted the boot process in order to correct the booting problem.

Appx1904(7:24-29).[6] In the Bramnick recovery process, a user manually edits computer files (i.e., the "boot registry") to correct problems. *Id.* In reading the claims on this manual process, the USPTO acknowledged the claims covered recovery involving human intervention.

The inventors did not disagree. They distinguished Bramnick as lacking recovery of "boot components," not as Bramnick requiring human involvement.

[6] Although Bramnick is not part of the record on appeal, counsel discussed it during argument on Apple's motion (Appx1016), the face of the patent lists it (Appx48), it appears in exhibits filed with the Court (Appx1470), and it is publicly accessible. Rembrandt requests the Court take judicial notice of this patent. *See, e.g.*, *Hoganas AB v. Dresser Indus., Inc.*, 9 F.3d 948, 954 n.27 (Fed. Cir. 1993) (taking judicial notice of a patent not submitted to the trial court but cited in the file history of the patent-in-suit).

Appx1476-1477. The claims of the ’678 Patent issued without amendment on this issue, reflecting the understanding of the USPTO and the inventors that the claimed recovery encompassed human intervention.

4. The claims do not mention or suggest “automatic” recovery.

The ’678 Patent includes both system and method claims. Rembrandt asserted apparatus claims 1 and 3, and method claims 4 and 7 against Apple. None contains the word “automatically” or any synonym. Claim 1 only specifies that failed components are “recovered.” Appx72 (“wherein integrity failures are recovered through said trusted repository”). Claim 3, which depends from claim 1, adds that the trusted repository is a host computer. *Id.*

Method claim 4 requires neither the use of a trusted repository nor an “automatic” recovery, but only “recovering” failed boot components. Appx72; Appx73 (certificate of correction). Claim 7, which depends from claim 4, adds that recovery uses a secure protocol to obtain a replacement component from a trusted repository. Appx72.

C. Apple's Accused Products Implement a Secure and Reliable Bootstrap Architecture.

1. Apple designed its secure boot and recovery process to foreclose human activity from jeopardizing the integrity of an accused device, like the claimed invention.

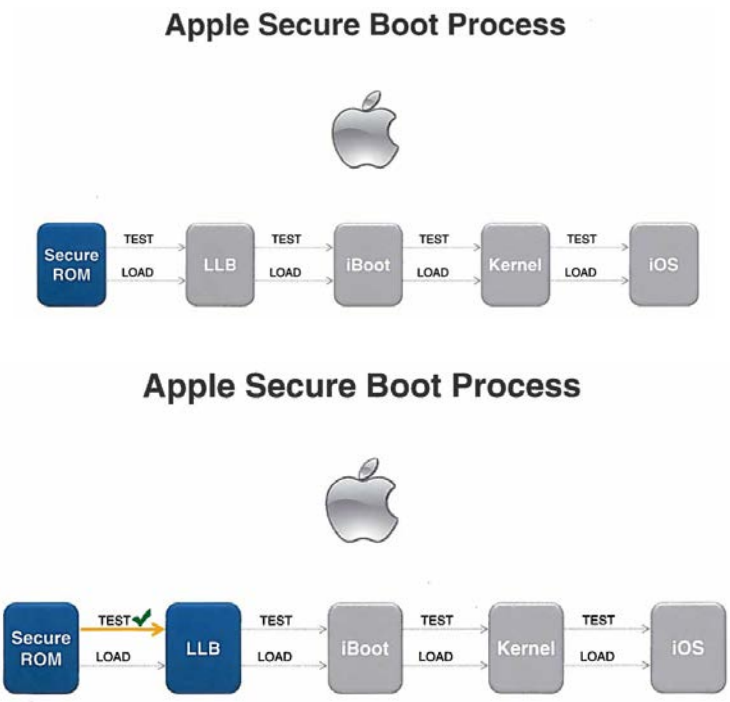
The district court described Apple's accused products as follows:

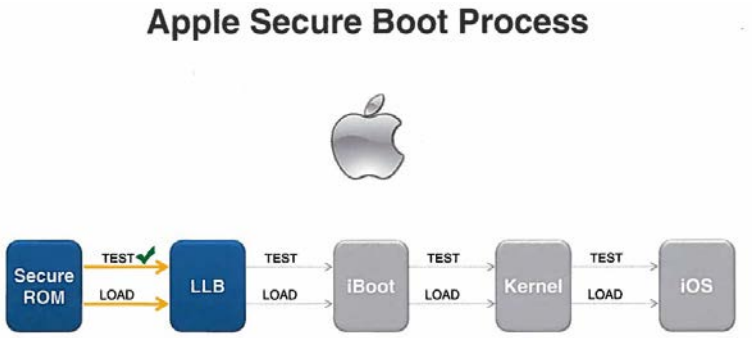
> The accused products are consumer mobile computing devices that run various versions of Apple's operating system for mobile devices, iOS. . . . Apple's products all use a chain of integrity checks upon booting. . . .
>
> Depending on which boot component caused an integrity failure, the accused products enter one of two modes reflecting the failure without any user interaction. The first is "Device Firmware Update" mode, and the second is "Recovery Mode." In either mode, a device can recover if it is connected to a computer that runs Apple's desktop software, iTunes. Once connected, iTunes is able to replace or repair the errant component; however, the user must affirmatively click a button to start the recovery process. If the user chooses to proceed with recovery, iTunes connects to a server maintained by Apple based on a network address hard-coded into the iTunes source code and retrieves information about the location of the desired replacement component. iTunes then uses that information to download the desired component. It then verifies the component, and if its integrity is verified, the component is loaded onto the device, and recovery is completed (Tygar Rpt. ¶¶ 112–24).
>
> A user's decision to forego or postpone the process will leave the phone "bricked," that is, useless until recovery can occur. . . . Once the user approves of the recovery, it proceeds without further human interaction (Tygar Rpt. ¶¶ 439–42, 454).

Appx5-6.

As the district court recognized, Apple's accused devices implement a secure boot process with layered integrity checks as in the '678 Patent. Appx1210. When a user activates an accused device, the device executes read-only SecureROM software in the processor of an accused device. Appx1209; Appx1765. That software is explicitly trusted because it cannot be altered after the processor has been created. *See* Appx1212; Appx1320-1323.

As with the '678 Patent, the SecureROM software cryptographically verifies the integrity of the low-level bootloader (LLB) before executing it. Appx1209; Appx1765. In the drawing below, Apple refers to the "verifying" step as a "test" and the "executing" step as a "load":





Appx1864-1866. The LLB cryptographically verifies the integrity of iBoot, and each component in the chain verifies the integrity of the next component before loading it. Appx1209-1211; Appx1765.

An integrity failure during this secure boot process causes the device to enter one of two states: Device Firmware Upgrade (DFU) Mode or Recovery Mode. Appx1214-1216. A device in either of these states can cause Apple iTunes to automatically obtain a suitable replacement component from Apple's servers, replace the failed component with a replacement component, and restart the device. *Id.*; Appx1218-1220; Appx1230-1231.

Apple's documents describe the automatic security features in the accused products, such as secure boot, as "largely invisible to the user, because security is handled by the system without the user's intervention." *See* Appx1238-1239 (citation omitted). Apple considered nonautomatic recovery procedures for the accused products, but rejected them. Appx1283. Apple decided to allow a user to "always be[] able to recover the device internally without requiring customer support help." *Id*. ("[W]e believe that it would be more convenient for our

customers or less frustrating for our customers if they were empowered to recover the devices themselves."); *see also* Appx1237-1238; Appx1360.

The connection of the accused product to a computer automatically launches iTunes (Appx1216-1217), which presents one of the following dialog boxes:

(Appx1217-1218). Unless users wish to maintain their device as a brick (i.e., in an unusable state), they must select either "OK" or "Restore" to begin recovery.[7]

[7] Apple says the product is "bricked" because it is no more functional than a brick. Appx1210-1211; Appx1230

Appx1211; Appx1218. After the user chooses to proceed, the recovery process completes automatically. Appx1216-1220; Appx1265-1266; Appx1269.

According to Apple's expert, Dr. Jaeger, the recovery process starts when the user chooses to proceed, and no further user intervention is required after that. Appx1274; Appx1277-1278. Apple's corporate witness agreed, explaining "in Apple's view, the beginning of Recovery is when the user chooses, in iTunes, to begin a software restore of an attached device." Appx1266. That witness also explained, "In the context of the accused products where a device had failed to boot and the user had chosen to restore the device via iTunes, at no point after that does the user interact with iTunes, nor the device." *Id.*; *see also* Appx1265.

Once recovery starts, iTunes determines the needed replacement components, locates and obtains them, and forwards the replacement components to the accused products. Appx1213-1214; Appx1218-1220; Appx1230-1232. The accused device replaces failed boot components with replacement components and reboots to restore the device. Appx1230-1232. This entire process occurs automatically, without user intervention. Appx1235-1240.

The user's connection of the accused device to a computer running iTunes and election to proceed occur before recovery begins. Those actions do not compromise the recovery process because the user does not replace the failed boot components (Appx1235-1240), and cannot select which components to replace

(Appx1277-78), unlike the recovery procedure the ’678 Patent criticizes (Appx63(3:45-54)).

2. Apple narrowed claims in its patents to avoid the ’678 Patent, and never distinguished the ’678 Patent as limited to recovery without human interaction.

Shortly before the launch of the Apple iPhone, Apple sought to patent the secure boot and recovery processes on that product. Appx1285-1286. The USPTO repeatedly rejected Apple’s applications over the ’678 Patent. *See, e.g.*, Appx1177; Appx1363. Apple did not distinguish the ’678 Patent as requiring automatic recovery without user intervention. *See, e.g.*, Appx1363-1364. Instead, Apple limited its patents in ways not relevant to the ’678 Patent, such as where to store the cryptographic signatures for layered integrity checks. Appx1364.

D. The District Court Limits the Term “Recovery” and Grants Summary Judgment.

Rembrandt filed suit in January 2014 in the Eastern District of Texas. Appx1648-1654. In October 2014, one week before the *Markman* hearing, that court transferred the case to the Northern District of California. Appx1684; Appx1688-1689.

1. Apple moves for summary judgment.

In May 2016, Apple filed its motion for summary judgment, arguing that each asserted claim requires automatic recovery, and that Apple’s products lack this feature because a user must plug the device into iTunes and choose to proceed

to recover from a failure. Appx1034-1041. Rembrandt opposed, explaining the inventors did not limit their invention to automatic recovery. Appx1150-1157. Rembrandt also provided undisputed evidence of literal infringement and infringement by equivalents even if the court adopted Apple's construction; the evidence included testimony by Apple witnesses, Apple documents, and opinions from Rembrandt's expert. Appx1157-1161.

Judge Alsup, who received the case, did not conduct a separate claim-construction hearing. Appx1704. Instead, following his procedure of resolving disputes over claim scope "on summary judgment or at trial in setting the jury instructions" (*id.*), he only addressed the single term, "recover[y]," because that was the only term at issue in the summary-judgment motion (Appx7; Appx1035).

Viewing the patent through the lens of "recovery," the court misguided its analysis of the invention toward automation, not security. Rembrandt was unable to correct this. During the hearing on summary judgment, the court gave Rembrandt fewer than ten minutes to convince it not to limit the claims to "automatic" recovery and grant Apple's motion for noninfringement. Appx1004-1005.

During the hearing, the district court described the specification as "soaked" with the term. Appx1004-1006. To give some perspective, the specification contains twenty-two columns of text, twelve drawing sheets, but only eleven

instances of "automated" or some variant (some are duplicative and the Order only cited six). [8] In contrast, the patent has more than 70 instances of "security" and its variant, including the title.

2. The district court grants summary judgment.

The next day, the district court granted Apple's motion for summary judgment, framing the issue as follows:

> The critical claim construction dispute here is whether the recovery element of the claims requires *automatic* recovery, that is, recovery without human intervention.

Appx7. The Order highlighted instances where the specification refers to automated recovery, the patent's criticism of prior-art recovery processes, and the two descriptions in the patent that refer to an embodiment's recovery process as occurring without user intervention. Appx8-9. The Order then made the factual finding "that the inventors *intended* for their invention to include an automatic recovery process." Appx9-10 (emphasis added).

[8] The term "automated" appears twice in the Abstract, once in the Field of the Invention, and once in the conclusion. Appx48; Appx62(1:22-25); Appx71(20:37-41). The term "automatic" or "automatically" appears seven times, some of which describe actions other than recovery. Appx63(3:57-59); Appx63(4:60-65); Appx64(5:3-5) (describing automatic updates); Appx64(5:48-51) (describing a preferred embodiment); Appx65(7:65-67) (describing a user's powering up a computer as "automatically" initiating a power on self-test); Appx71(20:45-48) (same as reference in column four); Appx71(20:54-56) (describing "automatic updates").

The Order also posited that failing to limit "recovery" as it did "might render the claims invalid under Section 112" because of "Rembrandt's failure to disclose an embodiment of the claimed invention involving a manual recovery process." Appx12-13. The Order provided no analysis on this point, and Apple never raised this defense (Appx1971-1972).

The Order disposed of Rembrandt's literal-infringement case without discussing the opinions of Rembrandt's expert or the evidence he used. Appx13-14. The Order also rejected Rembrandt's argument about the doctrine of equivalents, again without addressing the unrebutted opinions of Rembrandt's expert or the evidence he used. Appx14-16.[9]

IV. Summary of the Argument

The district court limited the claims of the '678 Patent by importing a limitation to require "automatic" recovery without human intervention, and erroneously granted summary judgment of noninfringement. In importing "automatic" the court found the inventors "implicit[ly]" disclaimed all nonautomatic recoveries. This was error because "recovering" and "recovered" are broad terms the inventors never limited, and the specification and prosecution

[9] Apple waived this issue because it did not address infringement under the doctrine of equivalents in its moving papers, nor did Apple's expert rebut Dr. Tygar's opinions on this issue. Apple only mentioned the issue in its reply (Appx1384-1385), after Rembrandt's opposition noted Apple's disregard of this issue (Appx1160-1161).

history not only lack any disclaimer of recovery involving a human, they embrace such recovery.

In granting summary judgment of noninfringement under the erroneous claim construction, the district court ignored evidence from Apple's corporate witnesses, Apple's engineers, Apple's documents, and Rembrandt's expert that the accused devices would infringe the asserted claims even if they required automatic recovery. Recovery begins only after a user connects the accused device to a computer running iTunes and chooses to begin the recovery process, which is how Apple's accused products operate. In granting summary judgment, the district court improperly resolved disputed issues of fact against Rembrandt without addressing them.

The district court similarly erred on the issue of infringement by equivalents. The only evidence of record came from Rembrandt's expert that the accused device infringed under the doctrine of equivalents. Apple offered no opinions from its own expert on this subject (Appx1031 n.2), relying instead on a single case involving different facts (Appx1384-1385).

V. Argument

A. The Standard of Review Is De Novo.

When the district court only used intrinsic evidence for its construction, as it did here, this Court reviews the claim construction *de novo*. *Teva Pharm. USA, Inc. v. Sandoz, Inc.*, 135 S. Ct. 831, 840-42 (2015).

A "'judge's function' at summary judgment is not 'to weigh the evidence and determine the truth of the matter but to determine whether there is a genuine issue for trial.'" *Tolan v. Cotton*, 134 S. Ct. 1861, 1866 (2014) (citation omitted). The evidence, and inferences drawn therefrom, must be viewed in the light most favorable to the opposing party. *Matsushita Elec. Indus. Co. v. Zenith Radio Corp.*, 475 U.S. 574, 587-88 (1986). This Court reviews summary-judgment decisions under the law of the regional circuit—here, the Ninth Circuit. *Ring & Pinion Serv. Inc. v. ARB Corp.*, 743 F.3d 831, 833 (Fed. Cir. 2014). The Ninth Circuit reviews summary-judgment decisions *de novo*. *Greater Yellowstone Coal. v. Lewis*, 628 F.3d 1143, 1148 (9th Cir. 2010).

"As a general rule, summary judgment is inappropriate where an expert's testimony supports the nonmoving party's case." *Southland Sod Farms v. Stover Seed Co.*, 108 F.3d 1134, 1144 (9th Cir. 1997) (citation omitted); *see also Pyramid Techs., Inc. v. Hartford Cas. Ins. Co.*, 752 F.3d 807, 818 (9th Cir. 2014) (stating that evidence is sufficient to raise a question of fact for the jury if a "reasonable

jury viewing the summary judgment record could find by a preponderance of the evidence that the plaintiff is entitled to a favorable verdict" (citation omitted)).

B. The District Court Erred in Adding "Automatically" to the Claims to Exclude Any Human Activity in Recovery.

1. The intrinsic record does not support limiting the claimed recovery elements to "automatic" recovery.

a. The Court should afford the inventors' chosen claim language its full scope.

The patentee is free to choose a broad term and obtain the full scope of its plain and ordinary meaning, absent any explicit redefinition or disavowal of that scope. *Thorner v. Sony Comput. Entm't Am. LLC*, 669 F.3d 1362, 1367 (Fed. Cir. 2012).

> If everything in the specification were required to be read into the claims, or if structural claims were to be limited to devices operated precisely as a specification-described embodiment is operated, there would be no need for claims. . . . It is the *claims* that measure the invention.

SRI Int'l v. Matsushita Elec. Corp. of Am., 775 F.2d 1107, 1121 (Fed. Cir. 1985) (en banc) (citation omitted).

The asserted claims require "recovering" a boot component. Appx72. They never use the word "automatic" or any synonym to describe the recovery. *Id.* The claims do not dictate how to recover a failed boot component, nor do they prohibit user involvement in that process.

Claims should receive the full scope of their chosen language, especially when the inventor knew how to draft more restrictive claims. *Acumed LLC v. Stryker Corp.*, 483 F.3d 800, 807 (Fed. Cir. 2007) (rejecting a narrow construction drawn from the specification when the inventor "chose a different term [for the claim] that implies a broader scope"). The inventors of the '678 Patent knew how to use "automatically" to modify "recovering" or "recovered," as they did when discussing certain embodiments. *See, e.g.*, Appx63(4:60-65). They chose not to.

Courts should not add modifiers, such as "automatically," to limit claims. *Johnson Worldwide Assocs., Inc. v. Zebco Corp.*, 175 F.3d 985, 989 (Fed. Cir. 1999) (refusing to construe "heading" as referring to the "direction of the trolling motor" and instead adopting the ordinary and accustomed meaning of "heading," which connotes only direction). "In short, a court must presume that the terms in the claim mean what they say, and, unless otherwise compelled, give full effect to the ordinary and accustomed meaning of claim terms." *Id.*

Nothing in the record shows the inventors viewed automated or automatic recovery as necessary to the invention. The '678 Patent focuses not on an absence of human activity but on the integrity of the bootstrap process.

b. A disclaimer must be clear and unmistakable.

Apple never argued that the patentees acted as their own lexicographers and defined "recovering" or "recovered" differently from their plain and ordinary

meanings. Apple instead based its construction on the notion of disclaimer, which requires "a clear and unmistakable disclaimer in the specification or the prosecution history." *Unwired Planet, LLC v. Apple Inc.*, No. 2015-1725, 2016 WL 3947839, at *3 (Fed. Cir. July 22, 2016) (citing *Thorner*, 669 F.3d at 1367). There is no such disclaimer in this case.

The standard for disclaimer of claim scope is "exacting." *Thorner*, 669 F.3d at 1366. "A disclaimer or disavowal of claim scope must be clear and unmistakable, requiring 'words or expressions of manifest exclusion or restriction' in the intrinsic record." *Unwired Planet*, 2016 WL 3947839, at *3 (citation omitted). "It is likewise not enough that the only embodiments, or all of the embodiments, contain a particular limitation." *Thorner*, 669 F.3d at 1366.

There is no disclaimer simply because the unmodified claims do not achieve every objective in the specification. "[T]he fact that a patent asserts that an invention achieves several objectives does not require that each of the claims be construed as limited to structures that are capable of achieving all of the objectives." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1327 (Fed. Cir. 2005) (en banc)) (quoting *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 908 (Fed. Cir. 2004)). In *Liebel-Flarsheim*, the district court required a "syringe receiving opening" to be in a pressure jacket, as it was in each disclosed embodiment. 358 F.3d at 904. Although the Abstract, Summary of the Invention, and all

embodiments "focus[ed] on the use of the invention in conjunction with pressure jackets," this Court rejected that construction, finding no language in the intrinsic record that "disclaim[s] the use of the invention in the absence of a pressure jacket." *Id.* at 908 ("Those passages, although focusing on the use of the invention in conjunction with pressure jackets, do not disclaim the use of the invention in the absence of a pressure jacket.").

"Every claim construction, and each potential disclaimer, has to be considered in the context of each individual patent." *Unwired Planet*, 2016 WL 3947839, at *3. In *Unwired Planet*, the dispute involved whether to limit "voice input" to transmission over a voice channel as opposed to a data channel. *Id.* at *2. Although the Summary of the Invention described the "translation process" as "establishing a voice communication channel," this Court refused to limit the claims to a "voice channel" because nothing in the intrinsic record "describing the translation process r[o]se to the level of 'manifest exclusion or restriction' of the claim scope." *Id.* at *3. That other claims expressly recited "establishing a voice communication channel" also showed that the patentee could have, but did not, limit the claims. *Id.* at *4.

c. The ’678 Patent does not provide a clear and unmistakable disclaimer of any human activity in the claimed recovery process.

The intrinsic record for this patent lacks the clear and unmistakable expression of manifest exclusion required to find disclaimer of any human activity in the claimed recovery process. The ’678 Patent refers only to “human interaction” or “user intervention” three times. Appx63(3:42-45); Appx64(6:24); Appx66(10:8). None rises to the level of a disclaimer of claim scope.

The first reference appears in the “Background of the Invention” section and explains, “Previous efforts to provide recovery of bootstrap components have required human interaction, typically to insert a floppy disk containing the new component or to boot from a floppy disk.” Appx63(3:42-45). These efforts allowed the user to perform recovery in a way that made the system insecure. The ’678 Patent criticized these recovery efforts because of no physical security for floppy disks and no verification of the software on the disks before use. Appx63(3:40-57). Neither amounted to a condemnation of all human activity.

The patent also noted that the prior-art procedure allowed the recovery of only a single boot component. Appx63(3:54-57). The concluding sentence of that paragraph states, “This is in contrast to the present invention which provides automatic recovery of *all* of the bootstrap components including ROM chips.”

Appx63(3:57-59) (emphasis added). The statement was just addressing the failure of the prior art to recover all components, not condemning human intervention.

Even if it were a broader comment, "[m]ere criticism of a particular embodiment encompassed in the plain meaning of a claim term is not sufficient to rise to the level of clear disavowal." *Thorner*, 669 F.3d at 1366 (citing *Epistar Corp. v. Int'l Trade Comm'n*, 566 F.3d 1321, 1335 (Fed. Cir. 2009)).

The second and third references, which are practically identical to each other, appear in the description of particular embodiments. They read, "Once the repair is completed, the system is restarted (warm boot) to ensure that the system boots. This entire [recovery] process occurs without user intervention." Appx66(10:6-8). The "warm boot" requires a human entering the ctrl-alt-del keys, (Appx65(8:3-5)), showing that the phrase "without user intervention" does not foreclose all human activity. Appx66(10:6-8).

The patent has no clear and unmistakable disclaimer of any human activity in the recovery process. The district court's citations do not show or suggest the need for no user intervention for the invention, and the specification lacks even a single statement limiting the claimed recovery to processes "without user intervention." No claims expressly require the absence of human intervention either.

d. The specification uses words of inclusion, not manifest exclusion as disclaimer requires.

The specification does not limit the term "recovery" to any particular embodiment. According to the Abstract, the invention "can" be augmented with automatic recovery (Appx48), meaning it is an option. The district court dismissed this statement but glossed over the three sentences of the passage it quoted on page 10 of the Order containing this reference:

> A major design decision is the consequence of a failed integrity check. A simplistic strategy is to simply halt the bootstrap process. However, the bootstrap process of the present invention *can be augmented* with automated recovery procedures which preserve the security properties of the bootstrap process of the present invention under the additional assumption of the availability of a trusted repository.

Appx10-11 (emphasis added) (quoting Appx48). The first sentence explains that responding to a failed integrity check presents a "major design decision." Appx48. The second sentence provides an option to halt the bootstrap process. *Id.* The third sentence provides an option at the other extreme: to augment "the present invention" with "automated recovery procedures" if a trusted repository is available. *Id.* Even in this extreme, the automated "procedures" still refer only to automated aspects of a recovery process rather than a completely automatic process.

The import of the third sentence is that automated recovery is not even an option if a trusted repository is not available. Claim 4, for example, recites "when

said boot component fails, recovering said failed boot component," but does not require a trusted repository. Appx72; Appx73 (certificate of correction). The requirement of a "trusted repository" appears in claim 7, which depends from claim 4, meaning claim 4 encompasses embodiments where recovery occurs without a trusted repository. Appx72.

The specification confirms that a trusted repository is not a requirement of claim 4. The specification provides that "if security is not a concern, then a less robust approach could be used." Appx63(4:56-59).[10] Recovery in claim 4 is not limited to any particular implementation, automatic or not.

The Summary of the Invention first provides that the "present invention" includes "a recovery process," rather than an "automatic" recovery process. Appx63(4:33-37). Other passages describe "automated" recovery as a feature enabled by the invention or as a possible application of the invention, not the invention itself. For example, automatic repair can be a feature of the invention:

> The present invention *can also* be utilized to reduce the Total Cost of Ownership (TCO) of a personal computer, through automatically detecting and repairing integrity failures, thereby permitting the user to continue to work without the nuisance of a trouble call to support staff and the associated down time.

[10] The patent also describes that one embodiment uses a "secure protocol" to inform a "trusted repository" that a failure has occurred and to obtain a valid replacement component (Appx63(4:48-51)), but claims 1, 3, or 4 do not require such a protocol (Appx72). Only claim 7 does (*id.*), further demonstrating the inventors intended to define their invention broadly.

Appx63(4:60-65) (emphasis added); *see also* Appx71(20:42-48). Automatic updating is another possible application of the invention:

> The present invention *also* enables the bootstrap components to be automatically updated.

Appx64(5:3-5) (emphasis added); *see also* Appx71(20:54-56).

When the patent describes the "Field of the Invention" as "relat[ing] to an architecture for initializing a computer system and more particularly to a secure bootstrap process and automated recovery procedure" (Appx62(1:22-24))—a sentence the district court read as definitional—the "more particularly" clause highlights an embodiment. It does not disclaim nonautomatic recovery procedures, and it certainly does not disclaim all human interaction in the recovery process. "[T]he fact that the inventor may have anticipated that the invention would be used in a particular way does not mean that the scope of the patent is limited to that context." *Northrup Grumman Corp. v. Intel Corp.*, 325 F.3d 1346, 1355 (Fed. Cir. 2003).

e. The district court's construction improperly excludes embodiments involving human activity in recovery.

When the district court wrote that it found nothing in the specification contemplating human intervention in recovery (Appx9), it overlooked several examples Rembrandt listed in its briefs. In one, the patent explains that when a

trusted repository is unavailable, the computer's further action "depends on the security policy of the user," which requires the user to choose how to proceed:

> In each case, AEGIS attempts to recover from a trusted repository, step **298**, as discussed below. Should a trusted repository be unavailable after several attempts, then the client's further action depends on the security policy of the user. For instance, a user may choose to continue operation in a limited manner or may choose to halt operations altogether.

Appx66(10:19-25). The Order dismisses this embodiment as outside the "invention" because the human action only happens "if the invention fails," and argues this activity does not describe recovery because the only options provided are "to continu[e] operation in a limited manner or halt[] operations altogether." Appx12.

The Order does not explain what it meant by the "invention fails." If the court meant "recovery" fails, it has made a finding unsupported by any evidence. The passage it was addressing simply provides examples of what to do when the trusted repository is not available during recovery. Appx66(10:19-25). Nothing forecloses a user from manually choosing to halt operations until a trusted repository becomes available to finish the recovery process, especially after user intervention (e.g., plugging in a network cable that became unplugged). *Id.* Continuing operation in a limited manner or waiting for the repository to become available allows the system to continue to operate with full system integrity.

The ’678 Patent also describes recovery where a system administrator schedules “hands on” repairs of ROM failures. Appx71(20:48-53). Even though such repairs are required for ROM failures, the patent describes this recovery as “automatic,” signifying that automatic[11] is not synonymous with the complete absence of human intervention. Rather, the patent uses the term “automatic” to describe the automation of certain aspects of a particular process. For example, the patent explains that “[a]utomatically detecting and repairing integrity failures permits the user to continue to work without the nuisance of a trouble call to the support staff and the associated down time spent waiting.” Appx71(20:45-48). Recovery of ROM chips (e.g., identifying a failed component, informing a system administrator about the failure, and scheduling the repair work) is still automated despite the need for manual activity: “A system administrator can monitor the log of the AEGIS trusted repository and identify those workstations that require ‘hands on’ repairs, e.g., ROM failure, and schedule the work to be done when the user is not using the computer.” Appx71(20:48-52).

The Order dismissed this embodiment by finding automatic recovery ends before repairs by a human (Appx12), but nothing in the specification supports this finding. A ROM is a boot component (Appx62(1:53-58)) whose integrity is

[11] “[T]he present invention . . . provides automatic recovery of all of the bootstrap components including ROM chips.” Appx63(3:57-59).

verified in the claimed architecture (Appx62(2:63-65); Appx63(3:57-59)). If a ROM fails its integrity check, its recovery requires human intervention (i.e., a hands-on repair). Appx63-64(4:65-5:3); Appx71(20:48-53). This human intervention occurs during, not after, recovery. Nothing in the specification suggests the system has "recovered" before replacing the failed ROM, as the Order suggests. Appx12.

f. The USPTO believed the claimed "recovery" included human activity, and the inventors never disagreed.

The USPTO originally rejected the claims of the '678 Patent over a recovery process requiring human interaction. Appx1469-1470; Appx1904(7:24-29). Although this recovery process was not automatic, the USPTO viewed it as within the scope of the claimed recovery (Appx1470), and the inventors tacitly agreed by not distinguishing the art on this basis (Appx1474-1477).

As in *Ventana Medical Systems, Inc. v. BioGenex Laboratories, Inc.*, 473 F.3d 1173, 1182-83 (Fed. Cir. 2006), the prosecution history of the '678 Patent supports a broad construction of the disputed term. *Ventana* considered whether to limit "dispensing" to "direct dispensing," where "the reagent is dispensed directly from the reagent container" rather than using an intermediate transport mechanism, such as a pipette. *Id.* at 1178 (citation omitted). During prosecution, the USPTO did not limit "dispensing" to "direct dispensing," stating "the process as claimed

can be practiced by another materially different apparatus or by hand, such as a manual pipette means." *Id.* at 1183. Nor did the patentee rely on "direct dispensing" to distinguish the cited prior art during prosecution. *Id.*; *see also Deere & Co. v. Bush Hog, LLC*, 703 F.3d 1349, 1358-59 (Fed. Cir. 2012) (rejecting patentee's proposed narrow construction of "rotary cutter deck" in part because it contradicted unchallenged rejections made during prosecution based on a broader construction).

In this case, as in *Ventana*, the USPTO did not limit the claims of the '678 Patent to any particular type of recovery, nor did the inventors use "automatic" recovery to distinguish the prior art. Because there has been no disclaimer, the Court should construe "recovering" and "recovered" without importing limitations.

2. Absent any disclaimer, it was error to import features from an embodiment into the claims.

Because a patent specification often describes specific embodiments of the invention, a court should not confine the claims to unclaimed features of those embodiments. *Ventana*, 473 F.3d at 1181 (quoting *Phillips*, 415 F.3d at 1323). "[P]atentees [are] not required to include within each of their claims all of [the] advantages or features described as significant or important in the written description." *Golight, Inc. v. Wal-Mart Stores, Inc.*, 355 F.3d 1327, 1331 (Fed. Cir. 2004). "It is . . . not enough [to import a limitation] that the only embodiments, or all of the embodiments, contain a particular limitation." *Thorner*, 669 F.3d at 1366.

In *Northrup*, the district court limited "bus interface unit" to use in a "command/response system," as the preferred embodiment did. The Court reversed and accorded the term its ordinary meaning even though (1) the inventor conceived the invention would be used principally, if not exclusively, in a command/response system; (2) the patent repeatedly referred to the advantages of the invention in a command/response system; and (3) the "Background of the Invention" and "Detailed Description of Preferred Embodiment" described command/response systems. *Northrup*, 325 F.3d at 1354-55.

None of these statements amounted to a disclaimer because the Court held the statements from the description of the preferred embodiment "are just that—descriptions of a preferred embodiment that operates in a command/response system" and "do not indicate that the invention can be used only with a 'command/response' protocol." *Id.* at 1355; *accord Brookhill-Wilk 1, LLC v. Intuitive Surgical, Inc.*, 334 F.3d 1294, 1301 (Fed. Cir. 2003) (refusing to limit "a remote location" to require the surgeon to be outside the operating room even though the single embodiment in the patent contemplated a surgeon outside the operating room).

In *ScriptPro LLC v. Innovation Associates, Inc.*, No. 2015-1565, 2016 WL 4269920 (Fed. Cir. Aug. 15, 2016), the claims covered an automatic dispensing system for prescription medication, and the Court considered whether to limit

those claims to systems that sort and store prescription containers by patient-identifying information, as the district court found. *Id.* at *3. On appeal, the patentee argued that the district court improperly focused on one purpose of the invention—to keep track of slot use by particular customers. *Id.* This Court agreed and reversed. It held, "Not every claim must contain every limitation or achieve every disclosed purpose." *Id.* at *4.

Although the specification only described embodiments that sorted and stored prescription containers by patient-identifying information, and criticized prior art that did not, the claims did not contain any such limitations. *Id.* at 8-9 (citation omitted). The patent instead explained containers can be stored "by patient, prescription, *or other predetermined storage scheme* without input or handling by the operator." *Id.* at 7 (citation omitted). "[A] specification's focus on one particular embodiment or purpose cannot limit the described invention where that specification expressly contemplates other embodiments or purposes." *Id.* at 8.

Although the '678 Patent describes automatic or automated recovery, these statements do not limit the claims because the inventors chose not to advance claims with these limitations. Rather, the specification simply explains that the invention allows automatic recovery. *See, e.g.*, Appx63-64(3:40-5:32).

The '678 Patent's primary solution to problems with prior-art architectures involves the integrity checks. *See, e.g.*, Appx62(1:64-2:1); Appx63(3:1-3, 4:33-

37); Appx64(6:7-16, 6:25-32). The inventors distinguished the prior art on this basis and left the recovery options unspecified in the claims. Nothing in the intrinsic record requires all recovery forego any user involvement.

3. The cases the district court cited involved explicit disclaimers or an inventor acting as his own lexicographer, neither of which apply to this case.

In finding the specification "implicit[ly]" disclaims recovery with human intervention, the district court relied on five cases, but none supports the existence of an "implicit disclaimer." One, *GE Lighting Solutions, LLC v. AgiLight, Inc.*, 750 F.3d 1304 (Fed. Cir. 2014), actually supports Rembrandt's claim construction. The district court limited "IDC connector" to the embodiments in the specification, even though the term had a well-established meaning in the art. *Id.* at 1308. This Court reversed and construed the term consistent with that well-established meaning, which was broader than the embodiments in the specification. *Id.* at 1308-10. "The standards for finding lexicography and disavowal are exacting," and "[i]t was error to import the structural limitations of the preferred embodiment and the structural limitations of the dependent claims into the term IDC connector." *Id.* at 1309-10.

Another case the district court cited, *Trustees of Columbia Univ. v. Symantec Corp.*, 811 F.3d 1359, 1365 (Fed. Cir. 2016), involved the inventor's definition of "byte sequence feature," a term the inventors coined in the specification. The

parties agreed the accused device did not analyze "machine code," so if "byte sequence feature" included only machine code, the accused devices did not infringe. *Id.* at 1366. The Federal Circuit held that "byte sequence feature" included only a machine-code representation because the specification and provisional application defined the term that way, and the patentee's construction would conflate "byte sequence feature" with "feature." *Id.* at 1365-66. This case does not involve a coined term.

Regents of University of Minnesota v. AGA Medical Corp., 717 F.3d 929 (Fed. Cir. 2013), is an express-disclaimer case. This Court affirmed the district court's construction of "first and second disks" as requiring two discrete disks because that construction reflected the plain language of the claims, and because the patentees distinguished single-disk prior art during prosecution. *Id.* at 935-37. The Court found the district court's construction to be "faithful to the ordinary meaning of the language of claim 1" because the dictionary definition of "affix," which the claims used to describe how the two disks were attached, required connecting two discrete objects. *Id.* at 937-38. Moreover, the specification disclosed various methods of affixing two discrete disks to one another, "indicating that the membrane disks forming the conjoint disk are structurally distinct." *Id.* at 936. By contrast, nothing in the plain language of the claims or the prosecution history of this case indicates an intent to limit "recovering" or "recovered."

Honeywell International, Inc. v. ITT Industries, Inc., 452 F.3d 1312, 1314 (Fed. Cir. 2006), is also an express-disclaimer case involving an electrical "'arcing' problem" with fuel filters having a polymer housing. This Court limited "fuel injection system component" to fuel filters because: (1) the problem the specification addressed was specific to fuel filters; (2) the specification described the invention as "relate[d] to a fuel filter for use in the fuel line that delivers fuel to a motor vehicle engine"; (3) the original title of the patent application was "Electrostatically Dissipative Fuel Filter"; and (4) a fuel filter was the only component in the specification that met the claim limitations. *Id.* at 1314-19 (citation omitted). In this case, the '678 Patent is not addressing the problem of automation, but rather a secure boot architecture that recovers failed components without undermining a system's integrity. The title says nothing about any "automatic" recovery, nor does the '678 Patent describe a system that only made sense without human involvement.

The fifth case the Order cites, *SciMed Life Sys., Inc. v. Advanced Cardiovascular Sys., Inc.*, 242 F.3d 1337 (Fed. Cir. 2001), provides the classic example of express disclaimer. The issue was whether to limit "catheter" to those with coaxial lumens or also include dual lumens. *Id.* at 1340. The Court held the patentee expressly disclaimed dual-lumen catheters because: (1) the specification described a coaxial configuration as the "basic sleeve structure for all embodiments

of the present invention contemplated and disclosed herein," (2) criticized dual-lumen catheters, and (3) touted advantages that only coaxial catheters could achieve. *Id.* at 1342-44 (citation omitted). The specification of the '678 Patent has no such words of manifest exclusion.

The district court's cases neither announce a new theory of "implicit" disclaimer nor compel the conclusion that "recovery" in the '678 Patent must be "automatic recovery." Unlike the cases in the Order, the '678 Patent describes embodiments of the invention with user intervention for recovery, which negates disclaimer.

4. Apple's untimely written-description argument lacks support.

The district court surmised the claims would be invalid under 35 U.S.C. § 112 if they covered nonautomatic recovery (Appx12-13) even though Apple did not make this argument in its opening brief and only made a passing comment on this issue in a footnote of its reply brief.[12] Appx1379 n.1. This footnote, filed more than two years into this case and after the close of fact and expert discovery, marked the first time Apple raised a written-description defense. Apple did not include the argument in its invalidity contentions (Appx1971-1972),

[12] Because Apple only raised this in its Reply, Rembrandt had no opportunity to brief the issue.

and none of Apple's experts offered any opinions on the matter, thus waiving this argument.

The court provided no analysis of this issue, ignoring this Court's rejection of "a regime in which validity analysis is a regular component of claim construction." *Phillips*, 415 F.3d at 1327. This Court limits such analysis to instances where the claim is ambiguous after applying all the tools of construction, and focuses on what the USPTO would have thought the claims meant. *Id.* at 1327-28. In this case, the USPTO thought the claims included recovery with user involvement. Moreover, because the '678 Patent expressly contemplates human interaction during recovery, there is written-description support for the broader claims. *ScriptPro*, 2016 WL 4269920, at *4.

5. The district court found facts that were not in the record.

The district court found, without citation, "Other problems, such as the down time associated with requiring human intervention, could only be solved with an entirely automatic recovery process." Appx10. The '678 Patent never makes this assertion and nowhere uses the phrase "entirely automatic" to characterize any recovery process. The '678 Patent instead describes instances where user activity is not only permitted, but required, such as performing a "warm boot" and hands-on repairs to recover a system with a ROM failure or when a trusted repository is not available. *See supra* at 11-13.

The only extrinsic evidence offered in connection with claim construction contradicted the court's observation. Dr. Tygar opined that a person of skill in the art at the time of the invention would not view the invention as requiring automatic recovery procedures.[13] Appx1224-1226; Appx1345-1346; Appx1349-1350; Appx1355. To the extent the district court relied on this fact which lacks support in the record, it erred. *Teva*, 135 S. Ct. at 836-37.

C. Even if the District Court's Construction Were Correct, the Court Erroneously Granted Summary Judgment of Noninfringement by Resolving Factual Disputes Against Rembrandt, the Nonmoving Party.

Infringement, both literally and under the doctrine of equivalents, is an issue of fact. *Absolute Software, Inc. v. Stealth Signal, Inc.*, 659 F.3d 1121, 1129-30 (Fed. Cir. 2011). Summary judgment of noninfringement is inappropriate when there is conflicting evidence on that subject. *See Ethicon Endo-Surgery, Inc. v. Covidien, Inc.*, 796 F.3d 1312, 1324-27 (Fed. Cir. 2015) (vacating summary judgment of no infringement where the district court improperly discounted the patentee's expert's testimony). Disputes regarding the application of construed claims to the accused products raise issues of fact. *Int'l Rectifier Corp. v. IXYS*

[13] Dr. Tygar, whose work the '678 Patent cited as the first presentation of a secure boot process, is a founder in the field of "secure human computation," which "look[s] at the role of human beings in security functions," and a recent paper of his in this field received the "Test of Time Award" at the USENIX Security Conference. Appx1345-1346.

Corp., 361 F.3d 1363, 1374-75 (Fed. Cir. 2004). In this case, there is at least a genuine issue of material fact whether the recovery process in Apple's devices is "automatic," especially since all the evidence of record supports this conclusion, so even if the Court agrees "recovery" means "automatic recovery," summary judgment is improper.

1. The unrebutted evidence shows Apple's iPhones, iPads, and iPod Touches have the recovery element of the claims in the '678 Patent under the district court's construction.

The district court characterized Rembrandt's literal-infringement argument as seeking "a second bite at claim construction by trying to ignore the automatic limitation." Appx13. It held as a matter of law that "[a]utomatic recovery simply cannot mean recovery started manually, even if the technical restoration of a new component is ultimately performed automatically (after a human has commenced the process)." *Id.* In making this finding, the district court cited nothing, and ignored evidence favorable to Rembrandt showing automatic recovery in the accused devices.

Rembrandt's opposition to Apple's motion provided ample evidence the accused products practice automatic recovery (Appx1157-1160), but the district court's Order addressed none of them.

a. Apple admitted that in its accused devices, recovery starts after a user decides to recover, and such recovery proceeds without user interaction.

Apple's corporate representative testified the accused devices start recovery "when the user chooses, in iTunes, to begin a software restore of an attached device." Appx1266. That witness even identified a point in the source code after the user's choice where the recovery begins and "any further input from the client is ignored." Appx1265-1266. The representative confirmed recovery occurs, from start to finish, without any user interaction: "In the context of the accused devices where a device had failed to boot and the user had chosen to restore the device via iTunes, at no point after that does the user interact with iTunes, nor the device." Appx1266. According to Apple, the recovery process ends when the system successfully reboots. Appx1265-1266.

b. Apple's accused products realize the advantages of the claimed recovery.

The '678 Patent explains the recovery process "permit[s] the user to continue to work without the nuisance of a trouble call to support staff and the associated down time." Appx63(4:60-65). Apple's engineers designed the accused devices to recover internally, without requiring help from customer support. Apple considered and rejected recovery processes requiring a trouble call to support staff, such as the recovery processes for Apple's computers:

> We had looked at how Macs don't really have recovery processes. How they attempt to be as fault tolerant as they can be across updates, but if an update fails, then there was often no way for the user to recover the device, and so that pushed us in a direction of always being able to recover the device internally without requiring customer support help.

Appx1283.

Apple documents described nonautomatic recovery causing a "bad customer experience preventing them to use [sic] the device" because the user would have to visit an Apple Store for recovery rather than using iTunes. Appx1360. An Apple engineer explained why this approach would create a poor customer experience:

> I have to get in my car, drive, for us here in the Valley, ten minutes; for somebody in India, days potentially, to have your device functional again. That would be a poor customer experience.

Id. The accused recovery processes allow the "bricked" devices to "recover themselves." *Id.*

c. Dr. Tygar opined that the recovery processes in the accused devices are "automatic."

Rembrandt's expert provided detailed opinions why the accused products perform automatic recovery when a user "confirms to proceed with the recovery," showing "the recovery is 'automatic' when a software component in the secure boot chain fails its integrity verification." Appx1231-1240; Appx1242-1249. He explained that, upon detection of an integrity failure, the accused products automatically prompt the user to connect the device to a computer with iTunes,

which then automatically launches iTunes and asks the user whether to proceed. Appx1229. If the user confirms, the claimed recovery process "proceeds without any user intervention until the product reboots and attempts to perform the secure boot process again." Appx1231-1232.

The Apple recovery process does not permit user intervention, and even ignores user input, during recovery. Appx1265. A user does not identify the failed boot component, locate a replacement component, obtain a replacement component, contact any Apple servers, verify the integrity of the replacement component, replace the failed component with the replacement component, or reboot the device after the failed component has been replaced. Appx1238-1239.

Apple argued that the user's connecting the accused device to a computer running iTunes and electing to proceed in iTunes renders the recovery nonautomatic and outside the scope of the '678 Patent. Appx1041. But Dr. Tygar opined that the accused recovery process is consistent with an embodiment in the specification:

> Consider the case of a user who recovers by connecting the product to iTunes and confirms to proceed with the recovery, rather than having Apple recover the product at a store. Even in this case, the recovery is "automatic" when a software component in the secure boot chain fails its integrity verification. First, because the product may not be connected to iTunes already at the time of an integrity failure, a trusted repository is unavailable. The '678 patent describes that when a trusted repository is not available, "the client's further action depends on the security policy of the user." '678 patent, 10:21-23. For the accused products, the policy is set to require the accused product

> to be plugged into a computer running iTunes and for a user to confirm to proceed with the recovery. These steps are consistent with permitting a user's security policy if the trusted repository is unavailable in the preferred embodiment.

Appx1236. The district court neither addressed nor accepted as true, as it should have under the summary-judgment standard, the opinions the accused devices infringe under Apple's construction that Dr. Tygar provided in his initial expert report, his reply report, and his deposition. Appx13-14.

d. An Apple user's activity before recovery is limited to trivial actions that do not jeopardize the integrity of the device.

The '678 Patent explains how a human's involvement in retrieving or replacing a boot component may pose a security risk (*see* Appx63(3:47-54)), which is why the patent describes embodiments that automate certain aspects of the recovery (e.g., the retrieval and replacement) (*see, e.g.*, Appx66-67 (10:47-11:14)).[14] Nothing in the patent states or implies that soliciting any user input after detecting an integrity failure (e.g., to determine how to proceed when the trusted repository is not available, to query when the computer will be available for hands-on repairs, or to require a user to press ctrl+alt+del) presents a security risk.

[14] Certain boot components, such as ROMs, require hands-on recovery because the software is contained in read-only memory. But even in these circumstances, the '678 Patent describes how the recovery of these components may be automated using a trusted repository. *See supra* 13.

Apple does not deny the retrieval and replacement of a failed boot component in the accused products is entirely automatic and without any user intervention. Rather, Apple argues that a user's performance of steps before recovery begins renders the recovery nonautomatic. This is inconsistent with the meaning of "automatic." An automatic transmission still requires a driver to engage the transmission, and an automated teller machine requires a customer to insert a card and provide certain information.

A user's initiation of the recovery process of Apple's products involves trivial actions, such as plugging in a device and telling iTunes to proceed. These do not jeopardize the integrity of the bootstrap process and do not render recovery nonautomatic.

2. The unrebutted evidence shows Apple's iPhones, iPads, and iPod Touches have the recovery element of the claims of the '678 Patent under the doctrine of equivalents.

Under the doctrine of equivalents, "a product or process that does not literally infringe upon the express terms of a patent claim may nonetheless be found to infringe if there is 'equivalence' between the elements of the accused product or process and the claimed elements of the patented invention." *Warner-Jenkinson Co. v. Hilton Davis Chem. Co.*, 520 U.S. 17, 21 (1997). The doctrine is founded on the principle that "if two devices do the same work in substantially the same way, and accomplish substantially the same result, they are

the same, even though they differ in name, form or shape." *Graver Tank & Mfg. Co. v. Linde Air Prods. Co.*, 339 U.S. 605, 608 (1950) (citation omitted). Equivalents is highly fact specific. *See, e.g.*, *Intendis GmbH v. Glenmark Pharm. Inc.*, 822 F.3d 1355, 1360-61 (Fed. Cir. 2016).

The district court held the accused devices did not perform recovery in the same way as the patented invention, but again ignored Rembrandt's contrary evidence.

a. A user's initiation of an automatic recovery process is substantially equivalent to an "entirely automatic recovery process."

Dr. Tygar opined that, if the claimed recovery limitation were not literally met, the Apple recovery process is equivalent to a completely automated process. Appx1239-1240; Appx1247-1248. He explained that the accused products perform the same or substantially the same function as "automating the recovery of an integrity failure," because user interactions do not change the function of automating the steps of recovering a software component that failed its integrity verification during the secure boot process. Appx1239-1240. He also explained that the accused products perform this function in the same or substantially the same way as the claimed invention because the actual recovery steps are automatic and the user is given a choice when the trusted repository is unavailable, consistent with the embodiments described in the '678 Patent. *Id.* Recovery in the accused

products occurs according to automated routines, and does so entirely without human involvement. *Id.*

Finally, Dr. Tygar explained how the accused products achieve the same or substantially the same result because integrity failures are recovered and the accused products can attempt to complete the secure boot sequence again. *Id.* "Requiring a user to plug a device into a computer running iTunes and confirm to proceed with a recovery is an insubstantial difference to requiring an 'automatic' recovery without involving plugging in the product or asking for the user's confirmation to proceed." Appx1239.

b. Apple did not contest Dr. Tygar's opinions on the doctrine of equivalents.

Apple failed to offer the district court any opinions from its expert under the doctrine of equivalents to rebut Dr. Tygar's analysis on that subject. Nor did Apple's motion for summary judgment even address Dr. Tygar's opinions under this doctrine. Appx1033-1041.

c. The district court's factual findings on equivalence are unsupported and improperly resolve disputes of fact against Rembrandt.

The Order sets out three problems the '678 Patent eliminated: (1) the possibility a user would misplace a backup component, (2) the possibility a user would install unsecured components, and (3) the need for a service call to overcome an inoperable device. Appx14. The Order did not base this list on

anything Apple provided, but instead fashioned it afresh, ignoring Dr. Tygar's opinions.

The Order also finds the accused devices still suffer from "some of" these problems because they are unusable "bricks" until recovered, users could uninstall iTunes from their computer, and users cannot restore the device without erasing their data unless they take their device to an Apple Store. Appx14-15. The court cited no basis for these findings, and Apple provided no supporting evidence.[15]

Contrary to the court's unsubstantiated findings, the record demonstrates Apple's recovery scheme solves these problems the same way as the invention. Because Apple's servers and iTunes have the replacement components, a user cannot misplace or tamper with them. Appx1239; Appx1246-1247. And if the Apple product becomes inoperable, Apple's recovery process fixes the problem without user intervention and without having to visit an Apple Store. Appx1244-1245; Appx1283. If the Apple product needs a hands-on repair, the user must schedule one. Appx1345. Nothing in the claims affects service calls. Besides, Apple's employees admit that the accused recovery is "always . . . able to recover

[15] The record rebuts the finding that "a user that seeks to recover his or her device *without* erasing his or her data must still endure the cost of a call or visit to the Apple store" (Appx15). *See, e.g.*, Appx1269. For example, a user may recover his or her device using iTunes in a way that will not erase their data. *Id.* (Apple witness describing how a user can recover using iTunes "without erasing the data, the user data that's on -- on the device").

the device internally without requiring customer support help," and the accused devices can "recover themselves," avoiding the need for a user to visit an Apple Store. Appx1283; Appx1360.

The only evidence about equivalents in the record resides in Dr. Tygar's report where, contrary to the district court's holding, he explained how the accused devices perform the same function, in the same way, to achieve the same result as "automatic" recovery. Appx1236-1240; Appx1247-1249.

d. The district court's reliance upon *Moore*, involving an express element and a term of degree, was misplaced.

The district court foreclosed Rembrandt's doctrine-of-equivalents argument by citing *Moore U.S.A., Inc. v. Standard Register Co.*, 229 F.3d 1091, 1106 (Fed. Cir. 2000). Appx15. The court misapplied that case because the claim term was "majority," a term of degree, which this Court held could not be equivalent to its direct opposite, "minority," without vitiating the meaning of the term. *Id.* at 1105-07.

In this case, "recovery" is not a term of degree and Rembrandt's proposed equivalence vitiates no express claim element.

Because the district court held the accused devices did not literally infringe because their recovery was not "entirely automatic," it concluded a user's initiation of an otherwise automatic process foreclosed infringement under the doctrine of

equivalents because it was not entirely automatic. "But courts have also recognized that to permit imitation of a patented invention which does not copy every literal detail would be to convert the protection of the patent grant into a hollow and useless thing." *Graver Tank*, 339 U.S. at 607. "Such a limitation would leave room for—and indeed encourage—the unscrupulous copyist to make unimportant and insubstantial changes and substitutions in the patent which, though adding nothing, would be enough to take the copied matter outside the claim, and hence outside the reach of [the] law." *Id.*

Apple has made an "unimportant and insubstantial" change—requiring a user to confirm to proceed with recovery after an integrity failure—and is engaged in precisely the conduct the doctrine of equivalents evolved to prohibit. *Id.* By finding no equivalence because of an inconsistency with the construed claims, the district court's holding defeats the entire purpose of the doctrine. "Failure of literal infringement is precisely the realm in which the doctrine of equivalents operates." *Schumer v. Lab. Computer Sys., Inc.*, 308 F.3d 1304, 1314 n.7 (Fed. Cir. 2002) (vacating summary judgment of no infringement because the district court conflated literal infringement and infringement under the doctrine of equivalents).

D. Rembrandt Requests Reassignment.

If the Court remands this case, Rembrandt respectfully requests reassignment to a different district judge. This Court defers to regional circuit law

on the issue of reassignment. *Eolas Techs. Inc. v. Microsoft Corp.*, 457 F.3d 1279, 1280-81 (Fed. Cir. 2006). The Ninth Circuit considers the following three factors for reassignment: "(1) whether the original judge would reasonably be expected upon remand to have substantial difficulty in putting out of his or her mind previously-expressed views or findings determined to be erroneous or based on evidence that must be rejected, (2) whether reassignment is advisable to preserve the appearance of justice, and (3) whether reassignment would entail waste and duplication out of proportion to any gain in preserving the appearance of fairness." *McCalden v. Cal. Library Ass'n*, 955 F.2d 1214, 1224 (9th Cir. 1990) (quoting *Davis & Cox v. Summa Corp.*, 751 F.2d 1507, 1523 (9th Cir. 1985)). A finding of either of the first two factors alone is sufficient to support reassignment to a new district judge. *Nozzi v. Hous. Auth. of Los Angeles*, 806 F.3d 1178, 1204 (9th Cir. 2015). Judicial remarks may reveal bias "if they reveal an opinion that derives from an extrajudicial source" and do reveal bias when "they display a deep-seated favoritism or antagonism that would make fair judgment impossible." *Liteky v. United States*, 510 U.S. 540, 555 (1994). "This court considers a transfer request with great caution, and, in the absence of personal bias, would grant such a request only in 'unusual circumstances.'" *Research Corp. Techs. v. Microsoft Corp.*, 536 F.3d 1247, 1255 (Fed. Cir. 2008) (quoting *Davis & Cox*, 751 F.2d at 1523) (ordering reassignment).

This case presents such "unusual circumstances" where reassignment is advisable because the district judge appears to have difficulty in putting out of his mind previously-expressed views and, at the very least, to preserve the appearance of justice. The district judge's public statements reveal a hostility to non-practicing entities, and he called Rembrandt out as a non-practicing entity in his Order despite the lack of any relevance of that fact. The only explanation for mentioning this fact is that the district judge viewed Rembrandt's business model as relevant, which it should not be.

The district judge's feelings toward NPEs is not a secret. He issued an order quoting from an op-ed titled "Make Trolls Pay in Court" in a ruling against an NPE who sued forty companies. *Network Protection Scis., LLC v. Fortinet, Inc.*, No. C 12-01106 WHA, 2013 WL 4479336, at *1 (N.D. Cal. Aug. 20, 2013) (citation omitted). The quoted portion opines that when an NPE loses a case, "they are typically out little more than their own court-filing fees." *Id.*

In this case, the district judge appears to have difficulty setting aside his previous views towards non-practicing entities, such as the one from his prior case. He characterized Rembrandt's actions as being "just like those plaintiff lawyers that bring in 42 plaintiffs—42 patents and think they're going to try that case to a jury" (Appx2288), and that Rembrandt's attempt to amend its infringement

contentions to add Apple's newly released products was an attempt "to save the filing fee" (Appx2285).

The district judge's dislike of non-practicing entities also appeared to affect his handling of the substantive issues in the case. *See California v. Montrose Chem. Corp. of Cal.*, 104 F.3d 1507, 1521-22 (9th Cir. 1997) (reassignment is not warranted unless the district court's views towards a party appear to affect his or her handling of the substantive issues in the case). For example, he surmised defenses for Apple, offered advice on presenting the case to the jury, offered to give Apple more trial time, and offered to loan Apple's counsel the judge's personal computers for use as trial exhibits to present Apple's non-infringement case. Appx1747; Appx1759 (regarding an argument the Judge crafted, he told Apple's counsel: "That's a good argument. You should remember that. That would be a good argument for the jury."); Appx2288 ("Maybe they get 12 hours and you get 14.").

The district judge also appears predisposed towards ruling against Rembrandt. The court's summary-judgment Order castigates Rembrandt for failing to distinguish cases (Appx8-9) that Apple neither cited in its briefs (Appx1032-1038; Appx1378-1382) nor discussed at the summary-judgment hearing (Appx1000-1023). He also addressed a defense that was not in the case (Appx12-13 (addressing invalidity under § 112)), and granted summary judgment of

noninfringement under the doctrine of equivalents even though Apple never raised the defense in its moving papers. And he criticized in open court the lawyers who drafted the patent. Appx1005.

While the Court need not consider the third factor, *Nozzi*, 806 F.3d at 1204, it also weighs in favor of reassignment. Before entry of summary judgment, fact and expert discovery had been closed, the parties were preparing (but had not submitted) their pretrial filings, and the case was within a month of trial. Although the district judge conducted a one-hour technology tutorial, he has not otherwise conducted a *Markman* hearing or issued a *Markman* order and indicated he would resolve the disputed constructions during trial. Appx1704. The only substantive order issued in this case was the one at issue in this appeal. If the case is remanded, all that remains is setting this case for a jury trial. While the district judge's acquaintance with pretrial matters might expedite the trial, any duplication of time would not be so great as to be "out of proportion to any gain in preserving the appearance of fairness." *United States v. Sears, Roebuck & Co.*, 785 F.2d 777, 781 (9th Cir. 1986) (citation omitted).

The Ninth Circuit has ordered reassignment where a district judge summarily ruled against a plaintiff, allowed a defendant to assert untimely arguments, and offered strategic advice to defendant's counsel on how to win the case. *United States v. Jacobs*, 855 F.2d 652, 656-57 (9th Cir. 1988). Here, as in

Jacobs, the actions of the district judge warrant reassignment. The district judge's prior statements and conduct in this case suggest he would have substantial difficulty in putting out of his mind previously-expressed views and, at the very least, present an appearance of bias. Rembrandt respectfully requests reassignment to maintain the appearance of fairness.

VI. Conclusion

For the foregoing reasons, Rembrandt respectfully asks the Court to reverse the district court's holding of "implicit disclaimer" and confirm "recover[y]" in the claims should not be limited by "automatically." Rembrandt also asks the Court to reverse the district court's grant of summary judgment, premised on its holding of implicit disclaimer, and to remand with instructions that the case be reassigned.

Date: September 26, 2016

Respectfully submitted,

/s/ E. Robert Yoches

E. Robert Yoches
FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, LLP
901 New York Avenue, NW
Washington, DC 20001-4413
(202) 408-4000

Jacob A. Schroeder
FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, LLP
3300 Hillview Avenue
Palo Alto, CA 94304-1203
(650) 849-6600

Attorneys for Plaintiffs-Appellants
Rembrandt Patent Innovations, LLC
and Rembrandt Secure Computing, LP

Certificate of Service

I certify that on September 27, 2016, this CORRECTED BRIEF OF PLAINTIFFS-APPELLANTS REMBRANDT PATENT INNOVATIONS, LLC AND REMBRANDT SECURE COMPUTING, LP was filed electronically using the CM/ECF system and served via the CM/ECF system on counsel for Patent Owner-Cross Appellant as follows:

Mark S. Davies
Orrick, Herrington & Sutcliffe LLP
1152 15th Street NW
Washington, DC 20005-1706

/s/ *Jacob A. Schroeder*
Jacob A. Schroeder

Certificate of Compliance

I certify that this CORRECTED BRIEF OF PLAINTIFFS-APPELLANTS REMBRANDT PATENT INNOVATIONS, LLC AND REMBRANDT SECURE COMPUTING, LP contains 13,378 words as measured by the word-processing software used to prepare this brief.

/s/ *Jacob A. Schroeder*
Jacob A. Schroeder

Addendum

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

REMBRANDT PATENT INNOVATIONS, LLC, and REMBRANDT SECURE COMPUTING, LP,

Plaintiffs,

v.

APPLE INC.,

Defendant.

No. C 14-05094 WHA (lead)
No. C 14-05093 WHA (consolidated)

JUDGMENT

For the reasons stated in the accompanying order granting summary judgment, **FINAL JUDGMENT IS HEREBY ENTERED** in favor of Apple Inc. and against Rembrandt Patent Innovations, LLC, and Rembrandt Secure Computing, LP. The Clerk **SHALL CLOSE THE FILE.**

IT IS SO ORDERED.

Dated: June 10, 2016.

WILLIAM ALSUP
UNITED STATES DISTRICT JUDGE

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

REMBRANDT PATENT INNOVATIONS LLC, and REMBRANDT SECURE COMPUTING, LP,

Plaintiffs,

v.

APPLE INC.,

Defendant.

No. C 14-05094 WHA (lead)
No. C 14-05093 WHA (consolidated)

ORDER ON CLAIM CONSTRUCTION AND CROSS-MOTIONS FOR SUMMARY JUDGMENT

INTRODUCTION

In this patent infringement action involving a secure boot sequence for a computer system, the parties seek construction of two related claim terms, defendant moves for summary judgment of non-infringement, and plaintiffs move to strike portions of defendant's expert's reply report and for partial summary judgment on defendant's invalidity counterclaim. For the reasons stated below, defendant's motion is **GRANTED**, and plaintiff's motion is **DENIED AS MOOT**.

STATEMENT

This action concerns the manner in which defendant Apple Inc.'s iPhone, iPad, and iPod Touch devices boot up. Plaintiffs Rembrandt Patent Innovations, LLC, and Rembrandt Secure Computing, LP (collectively, "Rembrandt") are non-practicing entities. Rembrandt Patent Innovations owns United States Patent No. 6,185,678, the sole patent asserted in this action.

Rembrandt Secure Computing is an exclusive licensee of the '678 patent and possesses the right to sue and recover for infringement thereof. Rembrandt acquired all rights to the '678 patent through a series of transactions with the inventors, the National Security Administration, and the University of Pennsylvania, all executed between 2011 and 2014.

The '678 patent is entitled "Secure and Reliable Bootstrap Architecture" and describes techniques for booting a computer system in a secure manner. The patented invention involves a "chain of integrity checks" beginning with the foundational layer of the computer system — certain important hardware and the Basic Input/Output System ("BIOS") — which is assumed to be secure. Relying on that assumption, once the computer is powered on, the BIOS verifies the integrity (that is, confirms it is safe to load) of the next layer in the boot sequence before passing control of the system to that layer (which repeats the process for the subsequent layer), a process referred to as "bootstrapping." Such integrity checks are generally performed using now-ubiquitous algorithms involving public-key cryptography and digital signatures, the details of which are unimportant for this motion. When integrity is verified at all layers, the operating system loads, and the user can be assured of the system's integrity (subject to the assumption of the integrity of the foundational layer).

The instant motion concerns how the invention handles integrity failures: "Once an integrity failure is detected, the invention uses a secure protocol to inform a trusted repository that a failure has occurred and to obtain a valid replacement component" ('678 patent col. 4:49–51). That is, the patented invention contemplates storing replacements for the various components of the boot sequence in a trusted location and using those backups to repair any integrity failures.

The disclosed bootstrapping process, including its recovery protocol, ensures integrity of a given operating system, but it "can also be utilized to reduce the Total Cost of Ownership (TCO) of a personal computer, through automatically detecting and repairing integrity failures," obviating the need for a "trouble call to support staff and the associated down time" ('678 patent, col. 4:60–65).

Neither the inventors nor Rembrandt ever developed a commercial product based on the '678 patent; however, the patent discloses various preferred embodiments in a system called AEGIS, intended to be used on the IBM PC architecture.

Rembrandt accuses Apple's iPhone, iPad, and iPod Touch products of infringing claims 1, 3, 4, and 7 of the '678 patent. Independent claim 1 reads (*id.*, cols. 21:39–22:11):

> An architecture for initializing a computer system comprising:
>
> a processor;
>
> an expansion bus coupled to said processor;
>
> a memory coupled to said expansion bus, said memory storing a system BIOS for execution by said processor upon power up of the computer system;
>
> a plurality of boot components coupled to said expansion bus and accessed by said processor when said system BIOS is executed;
>
> a trusted repository coupled to said expansion bus; and
>
> means for verifying the integrity of said boot components and said system BIOS wherein integrity failures are recovered through said trusted repository.

Claim 3 depends from claim 1 and reads (*id.*, col. 22:15–19):

> An architecture for initializing a computer system according to claim 1, wherein said trusted repository is a host computer communicating with said computer system through a communications interface coupled to said expansion bus.

Independent claim 4 reads (*id.*, col. 22:20–26):

> A method for initializing a computer system comprising the steps of:
>
> (1) invoking a Power on Self Test (POST);
>
> (2) verifying the integrity of a system BIOS;
>
> (3) verifying the integrity of a boot component; and
>
> (4) when said boot component fails, recovering said failed boot component.

Claim 7 depends from claim 4 and reads (*id.*, col. 22:37–40):

> The method of claim 4, wherein step (4) employs a secure protocol to obtain a replacement boot component from a trusted repository to replace said failed boot component.

The accused products are consumer mobile computing devices that run various versions of Apple's operating system for mobile devices, iOS. Rembrandt accuses every version of the iPhone, iPad, and iPod Touch and every version of iOS released through September 2015.*

Apple's products all use a chain of integrity checks upon booting. The similarities and differences between Apple's boot sequence and that of the '678 patent need not be addressed here. The key issue is how Apple's products recover upon an integrity failure.

Depending on which boot component caused an integrity failure, the accused products enter one of two modes reflecting the failure without any user interaction. The first is "Device Firmware Update" mode, and the second is "Recovery Mode." In either mode, a device can recover if it is connected to a computer that runs Apple's desktop software, iTunes. Once connected, iTunes is able to replace or repair the errant component; however, the user must affirmatively click a button to start the recovery process. If the user chooses to proceed with recovery, iTunes connects to a server maintained by Apple based on a network address hard-coded into the iTunes source code and retrieves information about the location of the desired replacement component. iTunes then uses that information to download the desired component. It then verifies the component, and if its integrity is verified, the component is loaded onto the device, and recovery is completed (Tygar Rpt. ¶¶ 112–24).

A user's decision to forego or postpone the process will leave the phone "bricked," that is, useless until recovery can occur. A user may elect to leave the phone in that state in order to wait until it can be brought to an Apple store in the hopes that alternative recovery procedures

---

* As this case progressed, the parties stipulated to five successive amendments to Rembrandt's infringement contentions, each time adding a set of newly-released Apple products. This action, therefore, concerns twenty-five accused products in the three product categories identified above. An order denied Rembrandt's motion to amend its infringement contentions a sixth time, the first such amendment that Apple opposed, which would have expanded the case to include four new hardware products and a new operating system, all released in fall of 2015 (Dkt. No. 130).

available at the store might not require erasing the data on the phone (while recovery via iTunes will generally erase data). Once the user approves of the recovery, it proceeds without further human interaction (Tygar Rpt. ¶¶ 439–42, 454).

Rembrandt commenced the first of two now-consolidated actions in the Eastern District of Texas in January 2014. The actions were transferred to the District Court here in San Francisco in November 2014 and subsequently consolidated. Apple now moves for summary judgment of non-infringement, and Rembrandt moves for partial summary judgment on Apple's counterclaim and defense of anticipation and obviousness. This order follows full briefing, a ninety-minute tutorial explaining the technology, and oral argument on the motions.

ANALYSIS

Apple moves for summary judgment on all claims based on the theory that the construction of two related claim terms will be dispositive. Rembrandt moves to strike portions of Apple's expert's reply report and moves for partial summary judgment on Apple's anticipation and obviousness claims based on that motion to strike.

This order begins with Apple's motion. A claim for infringement (or non-infringement) involves two steps. The first step is to construe the claims, which this order does only to the extent necessary to resolve the instant motions. The second step is to compare the properly construed claims to the accused device.

1. CLAIM CONSTRUCTION.

Apple's motion for summary judgment turns on the construction of two related claim terms, namely, the meaning of "recovered" in claim 1 (and dependent claim 3) and "recovering" in claim 4 (and dependent claim 7). The parties' proposed constructions are set forth below:

Terms & Claims	Apple	Rembrandt
"Wherein integrity failures are recovered through said trusted repository." (Claims 1 and 3)	"wherein replacement components are automatically obtained from a trusted repository."	"wherein replacement components are obtained from a trusted repository."
"when said boot component fails, recovering said failed boot component . . . to replace said failed boot component." (Claims 4 and 7)	"at the time boot component fails its integrity verification, automatically recovering said failed boot component . . . automatically replacing said failed boot component."	"when said boot component fails its integrity verification, recovering said boot component . . . to replace said boot component that failed its integrity verification."

The critical claim construction dispute here is whether the recovery element of the claims requires *automatic* recovery, that is, recovery without human intervention. Apple contends that when read in light of the specification, it is clear that a person of skill in the art would understand that the invention required automatic recovery. Rembrandt argues that the absence of any reference to automation in the claim language precludes such a limitation and that such a limitation may only be found by impermissibly importing the limitation from the specification.

To determine the proper meaning of a disputed claim term, we first look to the intrinsic evidence, that is, the record before the USPTO, the prior art, and the specification of the patent itself. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1317 (Fed. Cir. 2005) (*en banc*). Although the claims themselves define the scope of the patent, they must be read in the context of the entire patent, including the specification, which is "the single best guide to the meaning of a disputed term." *Id.* at 1315. Nevertheless, we must "avoid the danger of reading limitations from the specification into the claim." *Id.* at 1323. "In order to avoid importing limitations from the specification into the claims, it is important to keep in mind that the purposes of the specification are to teach and enable those of skill in the art to make and use the invention and to provide a best mode for doing so." *Ibid.*

Generally, the words of a claim are to be given their ordinary and customary meaning as understood by a person of ordinary skill in the art at the time of the invention, with two exceptions: "(1) when a patentee sets out a definition and acts as his own lexicographer, or (2) when the patentee disavows the full scope of a claim term either in the specification or during prosecution." *Thorner v. Sony Computer Entm't. Am., LLC*, 669 F.3d 1362, 1365 (Fed. Cir. 2012) (citations omitted). In some cases, "the specification may reveal an intentional disclaimer, or disavowal, of claim scope by the inventor. In that instance . . . the inventor has dictated the correct claim scope, and the inventor's intention, as expressed in the specification, is regarded as dispositive." *Phillips*, 415 F.3d at 1316. Such a disclaimer need not be expressly made. *See Trustees of Columbia Univ. in the City of New York v. Symantec Corp.*, 811 F.3d 1359, 1363 (Fed. Cir. 2016) (rejecting argument that a patentee must expressly define a term or expressly disclaim the full scope of a claim term).

Indeed, the Federal Circuit recently acknowledged a trend of decisions in which it found "disavowal or disclaimer based on clear and unmistakable statements by the patentee that limit the claims, such as 'the present invention includes . . .' or 'the present invention is . . .' or 'all embodiments of the present invention are . . . .'" *GE Lighting Solutions., LLC v. AgiLight, Inc.*, 750 F.3d 1304, 1309 (Fed. Cir. 2014), rehearing denied (June 17, 2014) (citing *Regents of Univ. of Minn. v. AGA Med. Corp.*, 717 F.3d 929, 936 (Fed. Cir. 2013); *Honeywell Intern., Inc. v. ITT Indus.*, Inc., 452 F.3d 1312, 1316–19 (Fed. Cir. 2006); *SciMed Life Sys., Inc. v. Advanced Cardiovascular Sys., Inc.*, 242 F.3d 1337, 1343–44 (Fed. Cir. 2001)).

From the very outset — in the description of the field of the invention — the patent contemplates an automated recovery procedure: "This invention relates to an architecture for initializing a computer system and more particularly to a secure bootstrap process and *automated* recovery procedure" ('678 patent, col. 1:23–25) (emphasis added).

Moreover, the patent criticizes the prior art as flawed because it relied on human interaction in order to recover (*id.*, col. 3:42–59) (emphasis added):

> Previous efforts to provide recovery of bootstrap components have required *human interaction*, typically to insert a floppy disk containing the new component or to boot from a floppy disk. There are several reasons why this recovery method is inferior to the present invention. The first is that providing physical security for the floppy disk is extremely difficult. Users can take the disks wherever they like, and do whatever they like to them. The major shortcoming, however, in only using a boot disk is that none of the firmware is verified prior to use. Thus, a user can add or replace expansion boards into the system without any security controls, potentially introducing unauthorized expansion cards. Additionally, these efforts have only focused on repairing a single component of the entire process, i.e., only repairing the boot block, or the BIOS but not both. *This is in contrast to the present invention which provides automatic recovery* of all of the bootstrap components including ROM chips.

The patent further notes, in the summary of the invention (*id.*, col. 4:60–65) (emphasis added):

> The present invention can also be utilized to reduce the Total Cost of Ownership (TCO) of a personal computer through *automatically* detecting and repairing integrity failures, thereby permitting the user to continue to work without the nuisance of a trouble call to support staff and the associated down time.

This advantage of the patented invention is echoed again in the description of the preferred embodiment, which states (*id.*, col. 20:45–48) (emphasis added):

> *Automatically* detecting and repairing integrity failures permits the user to continue to work without the nuisance of a trouble call to the support staff and the associated down time spent waiting.

Moreover, the description of the recovery process of the preferred embodiments *twice* states "[t]his entire process occurs without user intervention" (*id.*, cols. 6:25, 10:8).

References to automatic recovery pervade the patent. There is not a single reference to recovery with human intervention.

Rembrandt does not engage with the above-cited authority directing us to interpret claim terms in light of the specification and implicit disclaimers therein. Instead, it stands on the principle that "[i]t is the *claims* that measure the invention," not the specification. *SRI Intern. v. Matsushita Elec. Corp. of Am.*, 775 F.2d 1107, 1121 (Fed. Cir. 1985) (*en banc*). The above-cited authority is not at odds with that principle. As stated, the specification provides the best means for understanding the meaning of the claim terms. This order does not simply

import a claim limitation from the specification, but rather holds that the inventors intended for their invention to include an automatic recovery process, which was superior to a process involving human intervention.

At oral argument, counsel for Rembrandt contended that the description of the prior art that "required human interaction" cited above only disclaimed recovery using a *floppy disk*, not all recovery requiring human interaction. The specification simply does not support that reading. The prior art is described as requiring human interaction; the floppy disk is just one typical example of a means for recovery with human intervention. Counsel also argued that the floppy disk example served to distinguish a process that only recovers one component at a time, while the patented invention could recover all failed components at once, but that is only *one* of the problems with the prior art solved by the patented invention. Other problems, such as the down time associated with requiring human intervention, could only be solved with an entirely automatic recovery process. Rembrandt's attempt to limit the scope of the disclaimer in the specification is unpersuasive.

Rembrandt also contends that a genuine dispute of fact exists because its expert asserts that he would not have understood the term "recovery" to be limited to "automatic recovery" (Tygar Rebuttal Rpt. ¶ 42). "Legal error arises when a court relies on extrinsic evidence that contradicts the intrinsic record." *Ruckus Wireless, Inc. v. Innovative Wireless Solutions*, Nos. 2015-1425, 2015-1438, 2016 WL 3065024 (Fed. Cir. May 31, 2016) (citations omitted). This order declines Rembrandt's invitation to make such an error.

Rembrandt also argues that the patent discloses examples of manual recovery, but its examples are strained at best. Rembrandt cites the abstract of the patent, which notes "the bootstrap process of the present invention *can* be augmented with automated recovery procedures" (emphasis added). Rembrandt contends the use of the word "can" implies that automated recovery is not a necessary element of the invention. Reviewed in the context of the abstract, however, the sentence referenced does not relate to an augmentation *to the invention*, rather, it is a description of the invention itself:

> The basic principle is sequencing the bootstrap process as a chain of progressively higher levels of abstraction, and requiring each layer to check a digital signature of the next layer before control is passed to it. A major design decision is the consequence of a failed integrity check. A simplistic strategy is to simply halt the bootstrap process. However, the bootstrap process of the present invention can be augmented with automated recovery procedures which preserve the security properties of the bootstrap process of the present invention under the additional assumption of the availability of a trusted repository.

The sentences preceding the reference to automated recovery describe the bootstrap process aspect of the invention, and the problem encountered when an integrity check fails. The abstract then pivots from discussion of the bootstrapping elements of the invention toward the recovery component, thereby addressing the issue of failed integrity checks. Notably, that pivot is the first time recovery is mentioned anywhere in the patent. Thus, if the word "can" in the abstract were read to present an *option* of augmenting with automated recovery, there would be no recovery protocol at all, negating that claim element. This order rejects Rembrandt's proffered interpretation of the abstract.

Rembrandt makes a similar argument based on the statement in the specification that "[t]he present invention *can also* be utilized to reduce the Total Cost of Ownership (TCO) of a personal computer through automatically detecting and repairing integrity failures" ('678 patent col. 4:60–62) (emphasis added). The words "can also" in that sentence, however, are plainly intended to disclose the fact that the automatic recovery aspect of the invention can also reduce the total cost of ownership. It does not offer automatic recovery as an *optional* component of the invention. Again, Rembrandt's attempt to strain the language of the specification to fit its interpretation fails.

Rembrandt also points to a discussion of a preferred embodiment that contemplates resorting to a user policy upon an integrity failure, which might include human interaction (*id.*, col. 10:21–25) (emphasis added):

> Should a trusted repository be unavailable *after several attempts*, then the client's further action depends on the security policy of the user. For instance a user may choose to continue operation in a limited manner or may choose to halt operations altogether.

The user policy, however, is only reached if a trusted repository is unavailable "after several attempts." That is, we only ever reach the need for human intervention *if the invention fails*, *i.e.*, the automatic recovery fails. Moreover, the options available as part of that user policy do not include recovery but are rather limited to continuing operation in a limited manner or halting operations altogether. That embodiment simply cannot be read to enable a system that requires human intervention to recover.

Another embodiment provides for the intervention of a system administrator (*id.*, col. 20:48–53):

> A system administrator can monitor the log of the AEGIS trusted repository and identify the workstations that require "hands on" repairs, e.g., ROM failure, and schedule the work to be done when the user is not using the computer.

Again, the system administrator would only intervene once the system has contacted the trusted repository, automatically recovered (such that the user could continue to use the computer), and the log indicated that additional repairs must be done (although, again, the system *has* recovered).

Finally, Rembrandt points to a proposed use of the invention in which a system administrator can limit the validity time period of a given component, causing the system to contact the trusted repository on a set schedule to check for updates that might be placed there by the administrator. The manual intervention described in that embodiment, however, relates to the centralized management of updates and maintenance of the trusted repository. It does not disclose manual recovery from integrity failures. Indeed, the specification still describes the actual *update* process as occurring "automatically" (*id.*, col. 5:3–14) (emphasis added).

If Rembrandt's construction were to win the day, Rembrandt's failure to disclose an embodiment of the claimed invention involving a manual recovery process might render the claims invalid under Section 112. Claims "should be construed to preserve their validity." *Phillips*, 415 F.3d at 1368. This order finds a proper reading of the specification requires that the claim terms "recovered" and "recovering" must be qualified with the word "automatically,"

but the possibility that Rembrandt's construction would render the patent invalid lends further support to Apple's proposed construction.

In *Honeywell International, Inc. v. ITT Industries, Inc.*, 452 F.3d 1312, 1318 (Fed. Cir. 2006), the Federal Circuit noted "[t]he public is entitled to take the patentee at his word and the word was that the invention is [as described in the specification]." So too here. This order takes the patentees at their word that "this invention relates to . . . a secure bootstrap process and *automated* recovery procedure" ('678 patent, col. 1:23–25) (emphasis added).

2. NO LITERAL INFRINGEMENT.

Having construed the disputed claim term, we now turn to the accused products. Apple contends that because the accused products require human intervention to recover, they cannot infringe the asserted claims. Rembrandt responds that Apple's products *do* recover automatically *once the user has plugged the device into a computer and affirmatively initiated the recovery process* and because the devices *enter* "recovery mode" automatically (although, again, they cannot *recover* without human interaction). Rembrandt's argument seeks a second bite at claim construction by trying to ignore the automatic limitation. Automatic recovery simply cannot mean recovery started manually, even if the technical restoration of a new component is ultimately performed automatically (after a human has commenced the process). Indeed, the very language from the specification cited above plainly contemplates a recovery process that begins, proceeds, and finishes without user intervention. Further, the patent distinguishes a process (booting from a floppy disk) that must be *begun* manually, although it proceeds automatically once initiated.

True, the specification identifies certain circumstances in which human interaction may be necessary to recover a device, but, again, those circumstances are clearly limited to instances when several attempts at automatic recovery have *failed*, and in fact, they do not provide for recovery at all. Tellingly, no claim element reads on these instances of human interaction. Rather, they are merely demonstrations that when the invention malfunctions, the system is not irrevocably compromised.

Because the accused devices required human interaction to initiate the recovery process in the accused device, this order holds that the accused devices lacked an automatic recovery procedure, as required by each of the asserted claims.

3. NO EQUIVALENT INFRINGEMENT.

Rembrandt also contends that the accused products infringe under the doctrine of equivalents, which provides that "if two devices do the same work in substantially the same way, and accomplish substantially the same result, they are the same, even though they differ in name, form or shape." *Graver Tank & Mfg. Co. v. Linde Air Products Co.*, 339 U.S. 605, 608 (1950).

Rembrandt argues that Apple's recovery process still infringes because each accused device automatically detects a recovery failure and enters recovery mode. Then, once the user has plugged in the device and chosen to start the recovery process, the process continues automatically without human intervention. Because the user never has the opportunity to supply an unsecured component, the accused devices are assured a secure recovery, subject to the user's decision to plug in the device and initiate recovery. Contrary to Rembrandt, this cannot be read to recover a component that experienced an integrity failure "in the same way" as the patented invention.

The specification acknowledged that the invention's use of an automatic recovery process was intended to eliminate the possibility that a user would misplace a backup component, the possibility that he would install unsecured components, and the frustration and cost of making a service call and the intervening denial of service. It did so by removing the need for human intervention at all, namely, with automatic recovery. By contrast, the accused products require human interaction in order to recover, and as a result, some of the problems sought to be solved by the patented invention persist.

For example, it is undisputed that the accused products are "bricked" from the time they enter recovery mode until they are recovered, so a user will still experience a denial of service until he is able to reach a computer running iTunes or go to an iTunes store. Additionally, even

if remote, the possibility that the user would mistakenly uninstall or disable iTunes from his computer persists. Finally, a user that seeks to recover his or her device *without* erasing his or her data must still endure the cost of a call or visit to the Apple store. Thus, by adopting a different solution to one of the problems addressed by the patented invention, namely, assuring a secure recovery can occur for all compromised boot components, Apple forewent the solutions to other problems also addressed by the automatic recovery element of the invention. This can hardly constitute performance of the same function in the same manner.

The Federal Circuit rejected an argument similar to Rembrandt's in *Moore U.S.A., Inc. v. Standard Register Co.*, 229 F.3d 1091, 1106 (Fed. Cir. 2000), rehearing denied (Nov. 27, 2000). There, the patent owner claimed that the use of an adhesive strip on accused mailing forms that extended "about 48%" of the length of the margins of the mailer was equivalent to a claim that required materials to extend the "majority of the lengths" of the margins. That decision held that the patent owner's theory of equivalents would vitiate the "majority" claim limitation. If majority and minority were equivalent, the limitation would be unnecessary. Moreover, it held that no reasonable juror could find that "a minority — the very antithesis of a majority — could be insubstantially different from a claim limitation requiring a majority . . . ." *Ibid.*

Nor here. If Apple's recovery procedure that required human intervention could be equivalent to automatic recovery even though the former failed to address all of the problems solved by the latter, that would vitiate the automatic limitation present in the properly construed claims. Plainly, the inventors did not view these as equivalent when they expressly described previous efforts that required human interaction as "inferior to the present invention" ('678 patent, col. 3:43–47).

Rembrandt also argues that the recovery process on the accused devices is equivalent to the embodiments in the patent that present a user with a choice when a trusted repository is unavailable. Rembrandt ignores, however, the fact that the embodiments only present a user

choice *after several failed attempts* at automatic recovery, and moreover, that the user's choices presented *do not include recovery*.

Thus, this order holds Apple's products do not infringe under the doctrine of equivalents.

4. OTHER ISSUES.

Because this order holds that Apple is entitled to summary judgment, it need not address Apple's alternative arguments, namely, that it does not directly infringe because it does not sell its products tethered to a computer running iTunes, that it could not have acted willfully, and that it is not liable for extraterritorial damages under Section 271(f) for supplying a component from the United States for incorporation into an infringing product abroad.

This order also need not address Rembrandt's motion to strike portions of Apple's expert's reply report and to grant summary judgment that claims 1 and 3 are not anticipated or obvious in light of the "corresponding structures" identified in that reply report. Although Apple brings a counterclaim for declaratory judgment of invalidity, the district court has discretion to dismiss invalidity counterclaims after granting summary judgment that there is no literal or equivalent infringement, as here. *Nystrom v. TREX Co., Inc.*, 339 F.3d 1347, 1351 (Fed. Cir. 2003). The undersigned so exercises that discretion and dismisses Apple's counterclaim without prejudice.

CONCLUSION

To the extent stated above, Apple's motion for summary judgment of non-infringement is **GRANTED**. This order dismisses Apple's counterclaim for invalidity, and accordingly, Rembrandt's motion to strike Apple's expert's reply report and its motion for partial summary judgment is **DENIED AS MOOT**. Final judgment will follow.

IT IS SO ORDERED.

Dated: June 10, 2016.

WILLIAM ALSUP
UNITED STATES DISTRICT JUDGE

1955484

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office

August 12, 2014

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THIS OFFICE OF:

U.S. PATENT: *6,185,678*
ISSUE DATE: *February 06, 2001*

By Authority of the
Under Secretary of Commerce for Intellectual Property
and Director of the United States Patent and Trademark Office

P. R. GRANT
Certifying Officer

US006185678B1

(12) **United States Patent**
Arbaugh et al.

(10) **Patent No.: US 6,185,678 B1**
(45) **Date of Patent: Feb. 6, 2001**

(54) **SECURE AND RELIABLE BOOTSTRAP ARCHITECTURE**

(75) Inventors: **William A. Arbaugh**, Ellicott City, MD (US); **David J. Farber**, Landenberg; **Angelos D. Keromytis**, Philadelphia, both of PA (US); **Jonathan M. Smith**, Princeton, NJ (US)

(73) Assignee: **Trustees of the University of Pennsylvania**, Philadelphia, PA (US)

(*) Notice: Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: **09/165,316**

(22) Filed: **Oct. 2, 1998**

Related U.S. Application Data

(60) Provisional application No. 60/060,885, filed on Oct. 2, 1997.

(51) **Int. Cl.**[7] **G06F 9/00**; G06F 11/30

(52) **U.S. Cl.** **713/2**; 713/200

(58) **Field of Search** 713/1, 2, 100, 713/200, 201

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,200,770		4/1980	Hellman et al. .	
5,146,499	*	9/1992	Geffrotin	380/23
5,379,342		1/1995	Arnold .	
5,410,699	*	4/1995	Bealkowski et al.	713/2
5,421,006		5/1995	Jablon .	
5,564,054	*	10/1996	Bramnick et al.	713/2
5,629,980	*	5/1997	Stefik et al.	380/4
5,692,047	*	11/1997	McManis	380/4
5,745,669	*	4/1998	Hugard et al.	714/3
5,937,063	*	8/1999	Davis	380/4
5,974,546	*	10/1999	Anderson	713/2

OTHER PUBLICATIONS

J. Tygar & B. Yee, *Dyad: A System for Using Physically Secure Coprocessors*, Technical Report CMU-CS-91-140R, Carnegie Mellon University, May 1991.

B. Yee, *Using Secure Coprocessors*, Ph.D. Thesis, Carnegie Mellon University, May, 1994.

P. C. Clark, *BITS: A Smartcard Protected Operating System*, Ph.D. Thesis, George Washington University, May 8, 1994.

B. Lampson et al., *Authentication in Distibuted Systems: Theory and Practice*, ACM Transactions on Computer Systems, v10:265-310, Nov. 1992.

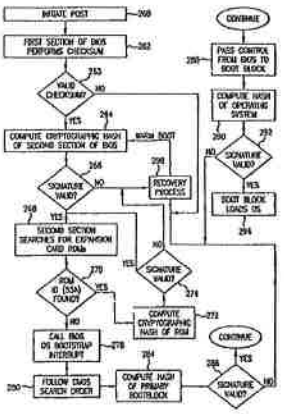
(List continued on next page.)

Primary Examiner—Dennis M. Butler
(74) *Attorney, Agent, or Firm*—Sterne, Kessler, Goldstein & Fox PLLC

(57) **ABSTRACT**

Integrity is rarely a valid presupposition in many systems architectures, yet it is necessary to make any security guarantees. To address this problem, the present invention discloses a secure bootstrap process, which presumes a minimal amount of integrity. The basic principle is sequencing the bootstrap process as a chain of progressively higher levels of abstraction, and requiring each layer to check a digital signature of the next layer before control is passed to it. A major design decision is the consequence of a failed integrity check. A simplistic strategy is to simply halt the bootstrap process. However, the bootstrap process of the present invention can be augmented with automated recovery procedures which preserve the security properties of the bootstrap process of the present invention under the additional assumption of the availability of a trusted repository. A variety of means by which such a repository can be implemented are disclosed with attention focused on a network-accessible repository. The recovery process is easily generalized to applications other than the bootstrap process of the present invention, such as standardized desktop management and secure automated recovery of network elements such as routers or "Active Network" elements.

7 Claims, 12 Drawing Sheets



OTHER PUBLICATIONS

R. Droms, *Authentication for DHCP Messages*, expired RFC draft, Nov. 1998.
D. Eastlake & C. Kaufman, *Domain Name System Security Extensions*, Internet RFC 2065, Jan. 1997.
W. Diffie et al., *Authentication and Authenticated Key Exchanges*, Codes and Cryptography, 2:107–125, 1992.
Digital Signature Standards, Technical Report FIPS–186, U.S. Department of Commerce, May 1994.
Secure Hash Standard, Technical Report FIPS–180–1, U.S. Department of Commerce, Apr. 1995.
HMAC: Keyed–Hashing for Message Authentication, Internet RFC 2104, Feb. 1997.
Dynamic Host Configuration Protocol, Internet RFC 2131, Mar. 1997.
DHCP Options and BOOTP Vendor Extensions, Internet RFC 2132, Mar. 1997.
J Reynolds & J Postel, *Assigned Numbers*, Internet RFC 1700, Oct. 1994.
K. R. Sollins, *The TFTP Protocol (revision 2)*, Internet RFC 1350. Jul. 1992.
G. Malkin & A. Harkin, *TFTP Option Extension*, Internet RFC 1782, Mar. 1995.

* cited by examiner

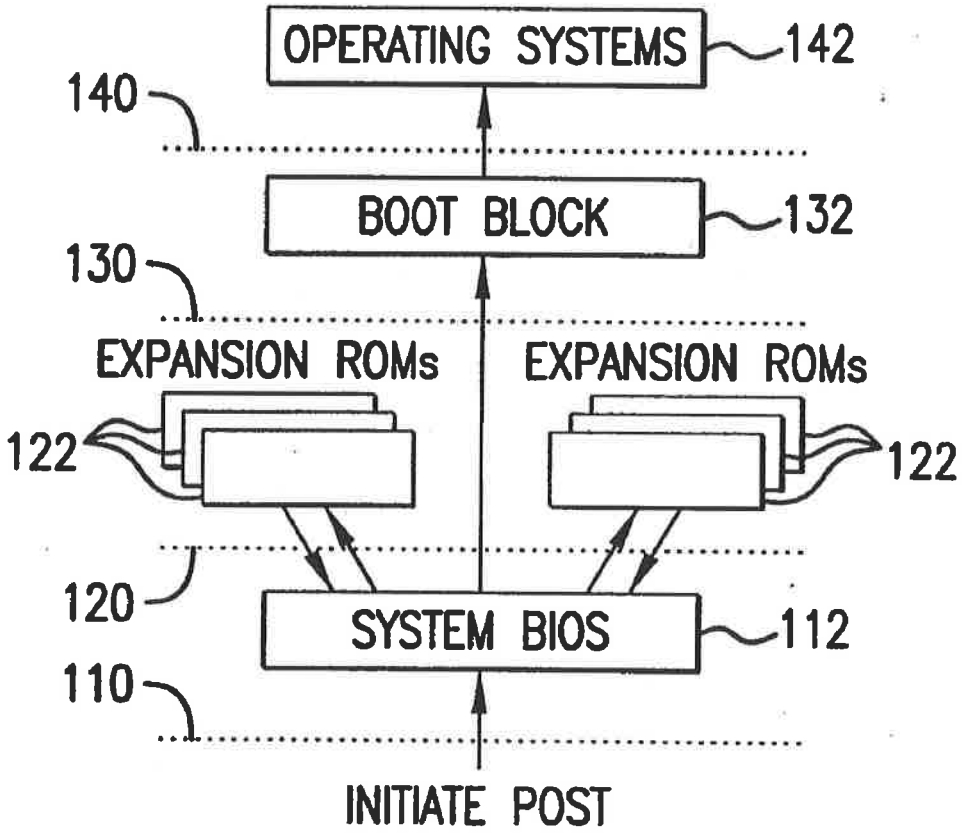


FIG. 1a

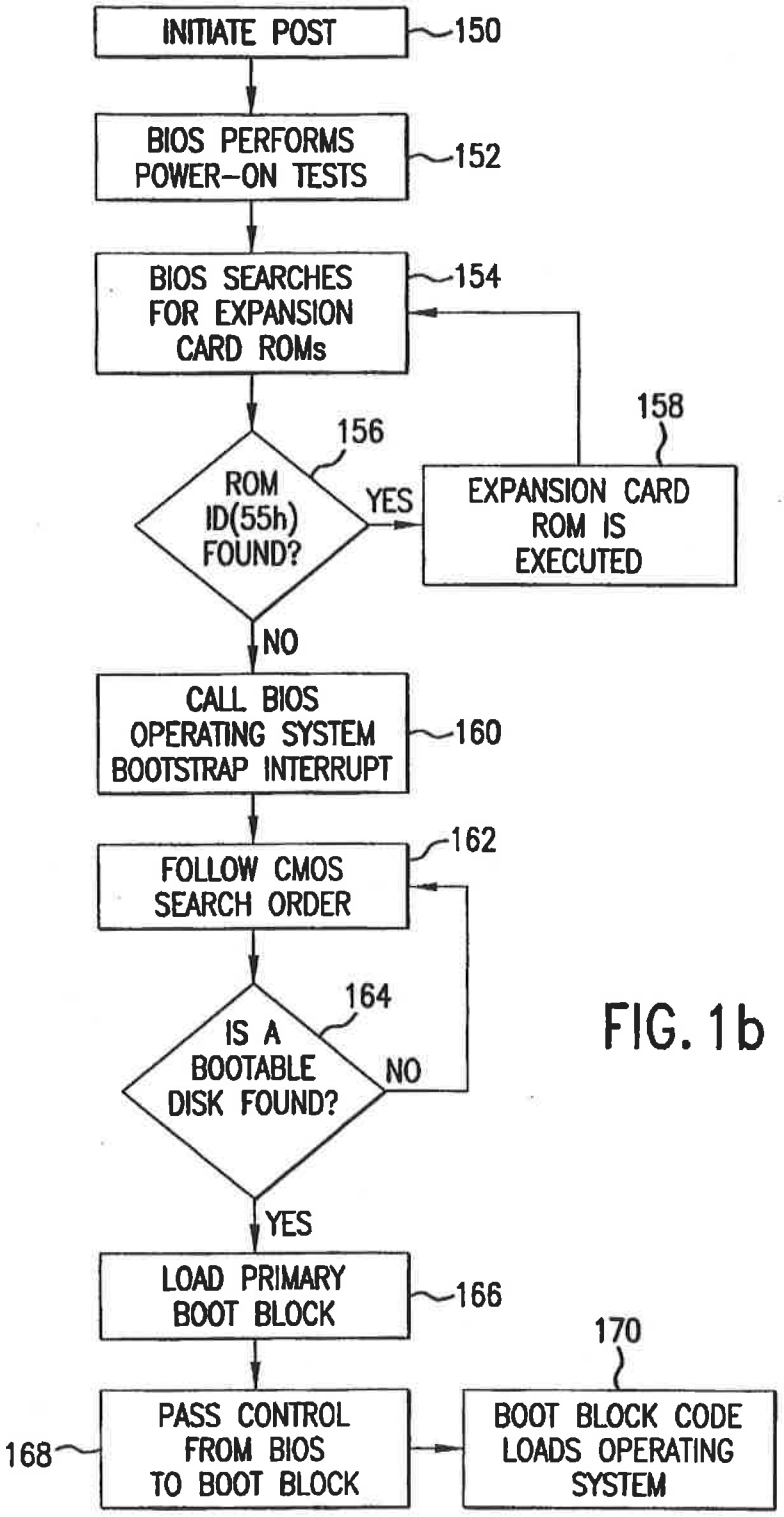


FIG. 1b

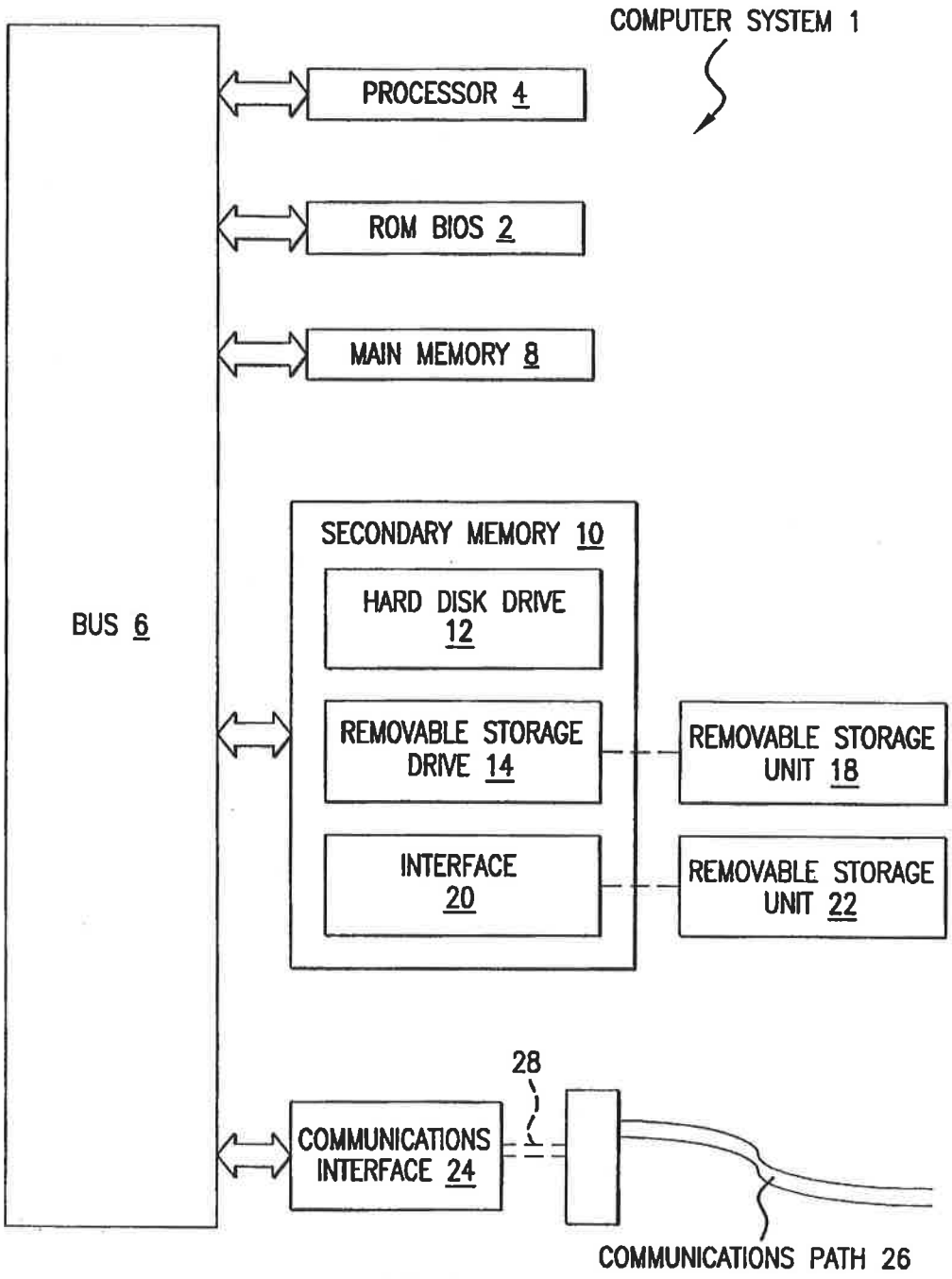


FIG. 1c

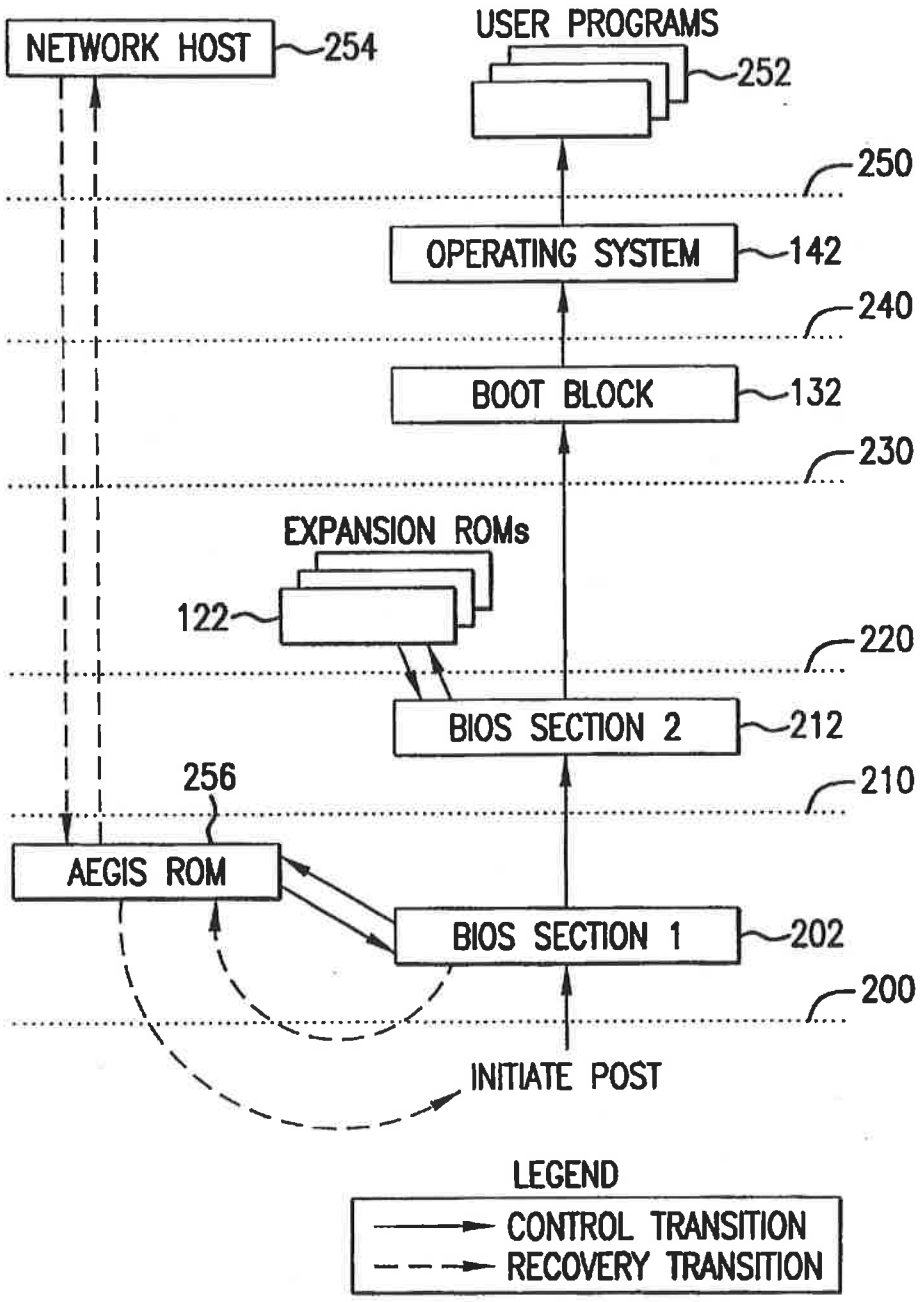


FIG. 2a

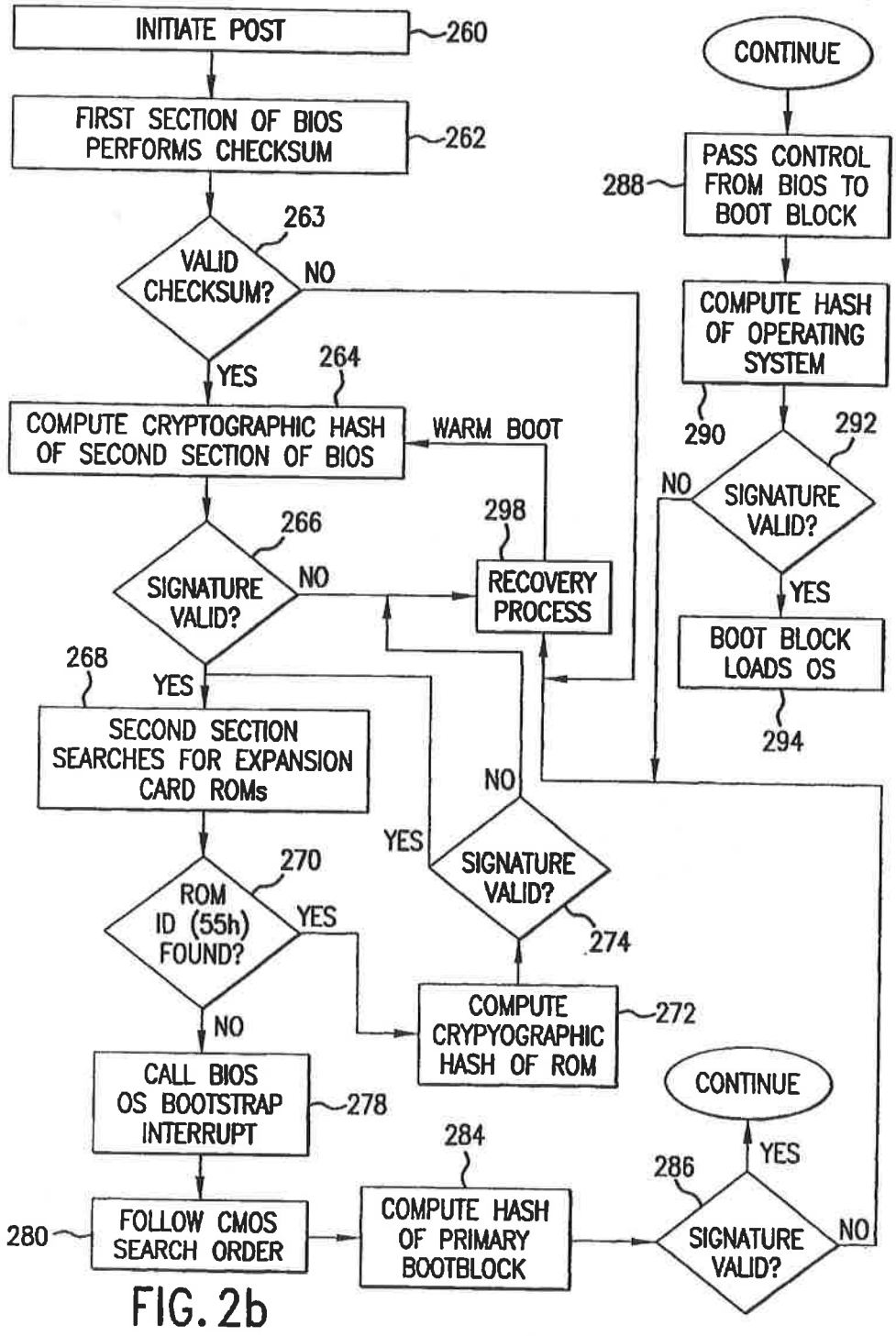


FIG. 2b

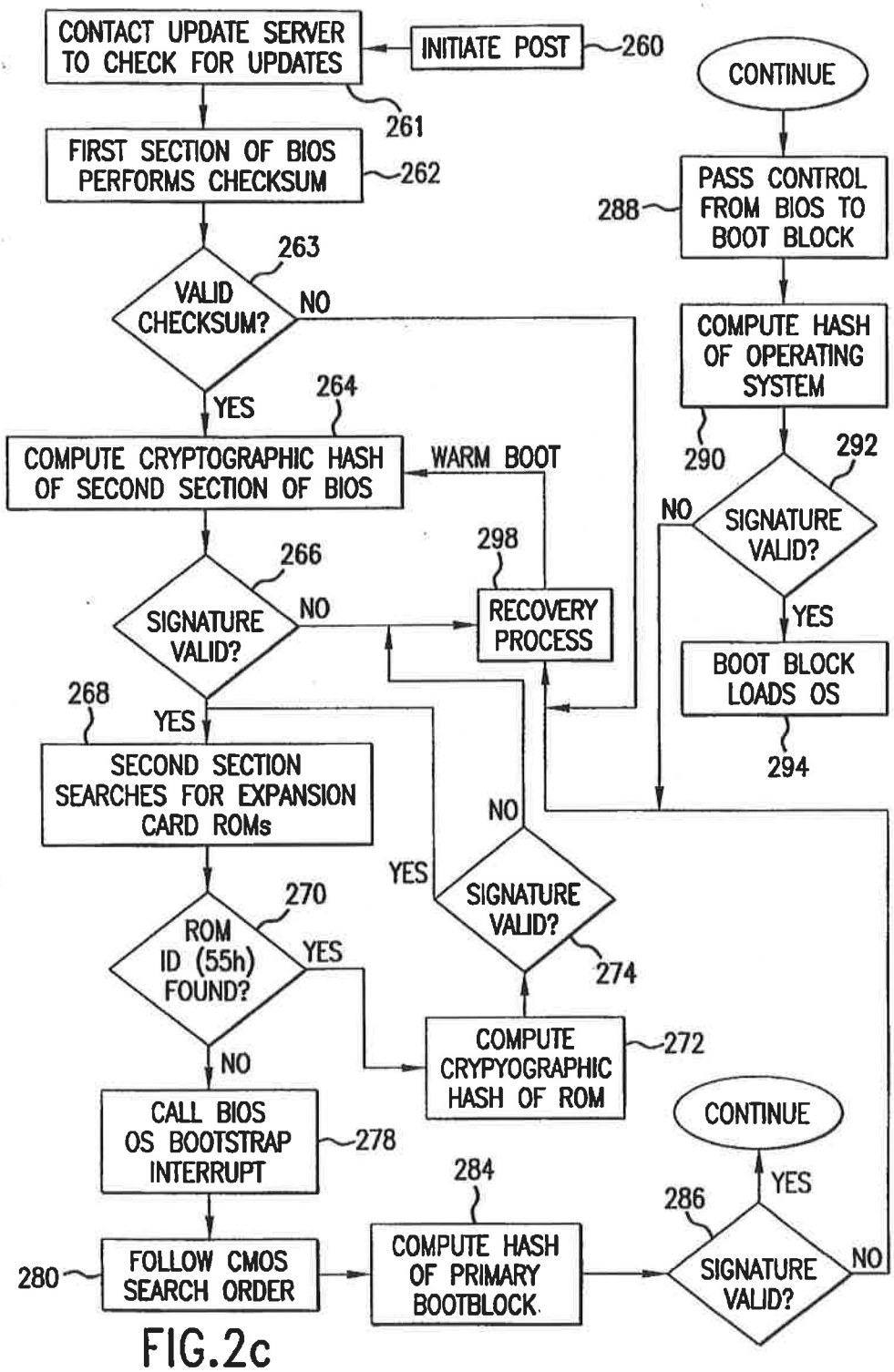


FIG.2c

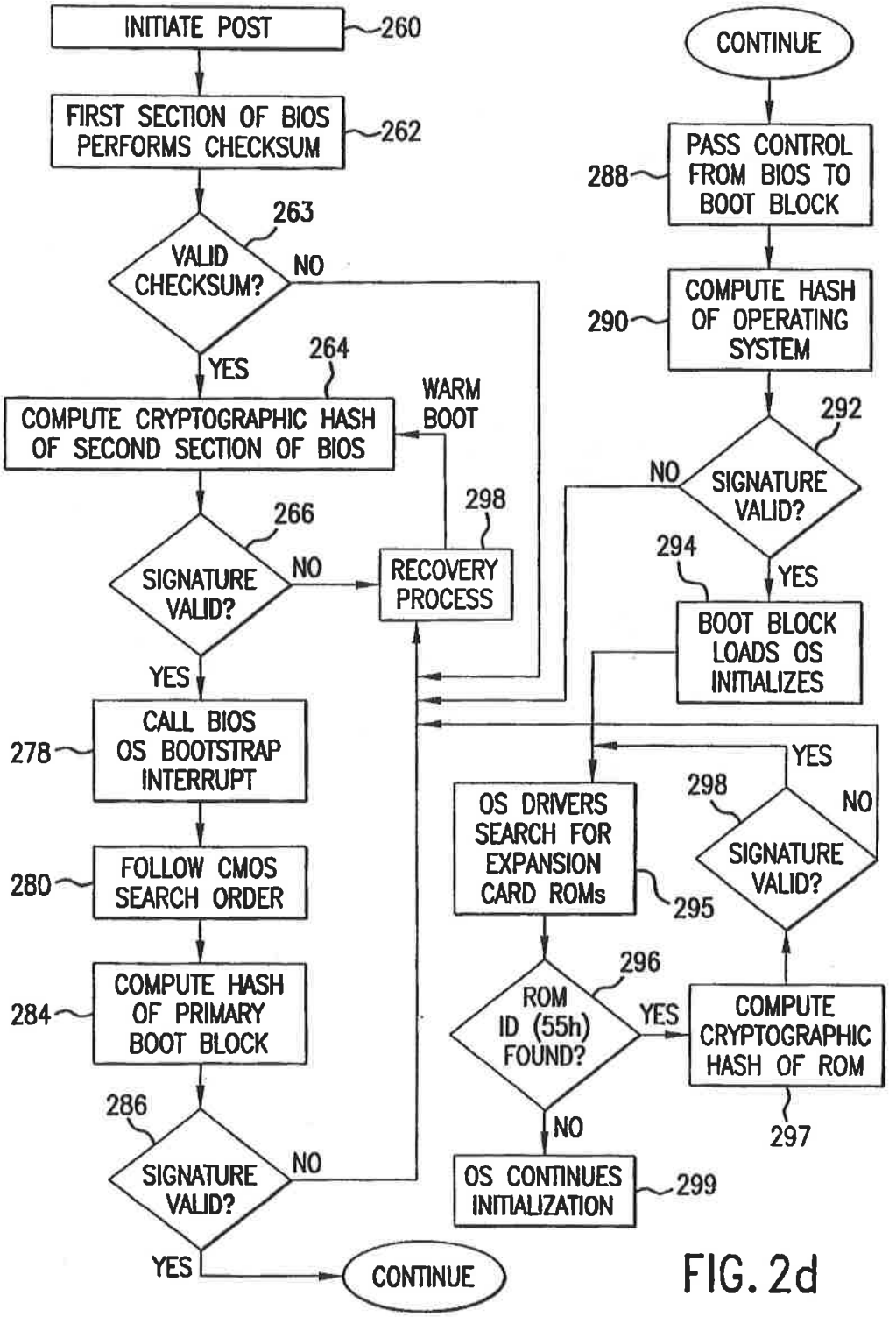


FIG. 2d

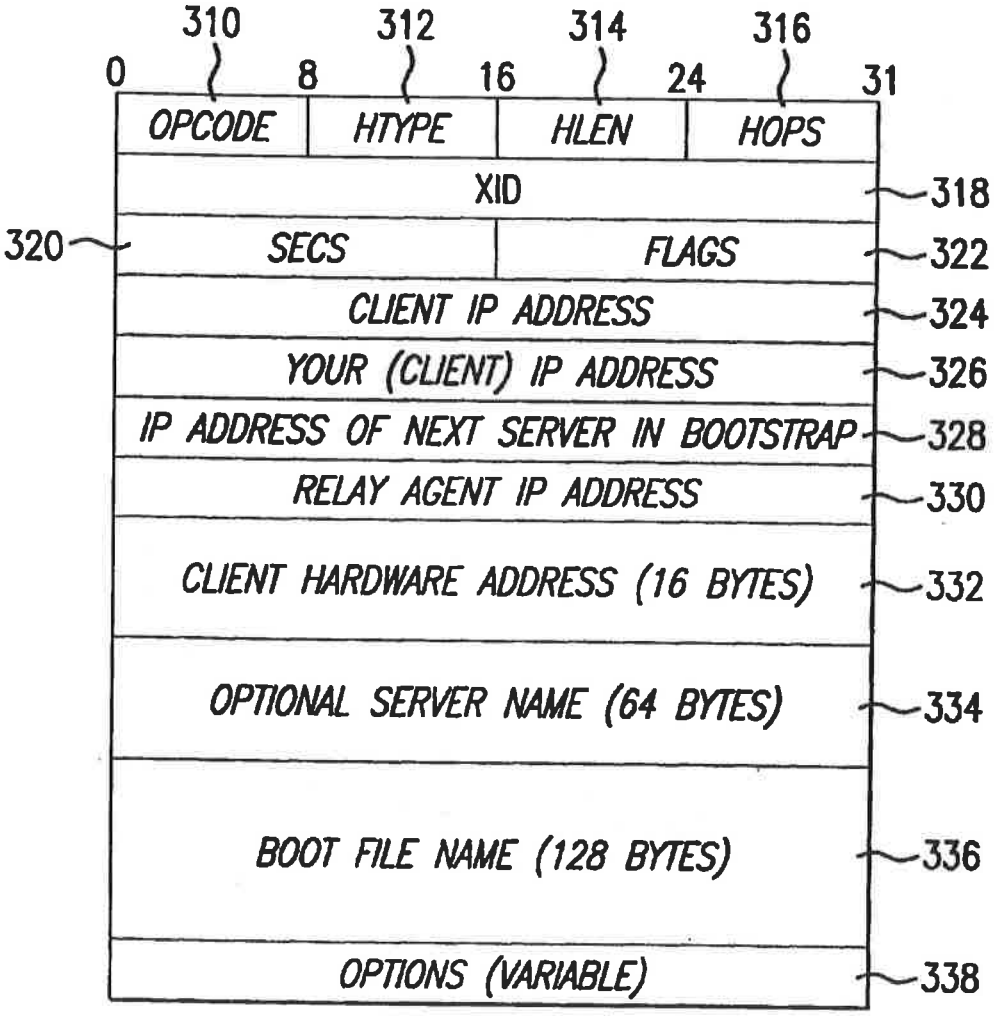


FIG.3

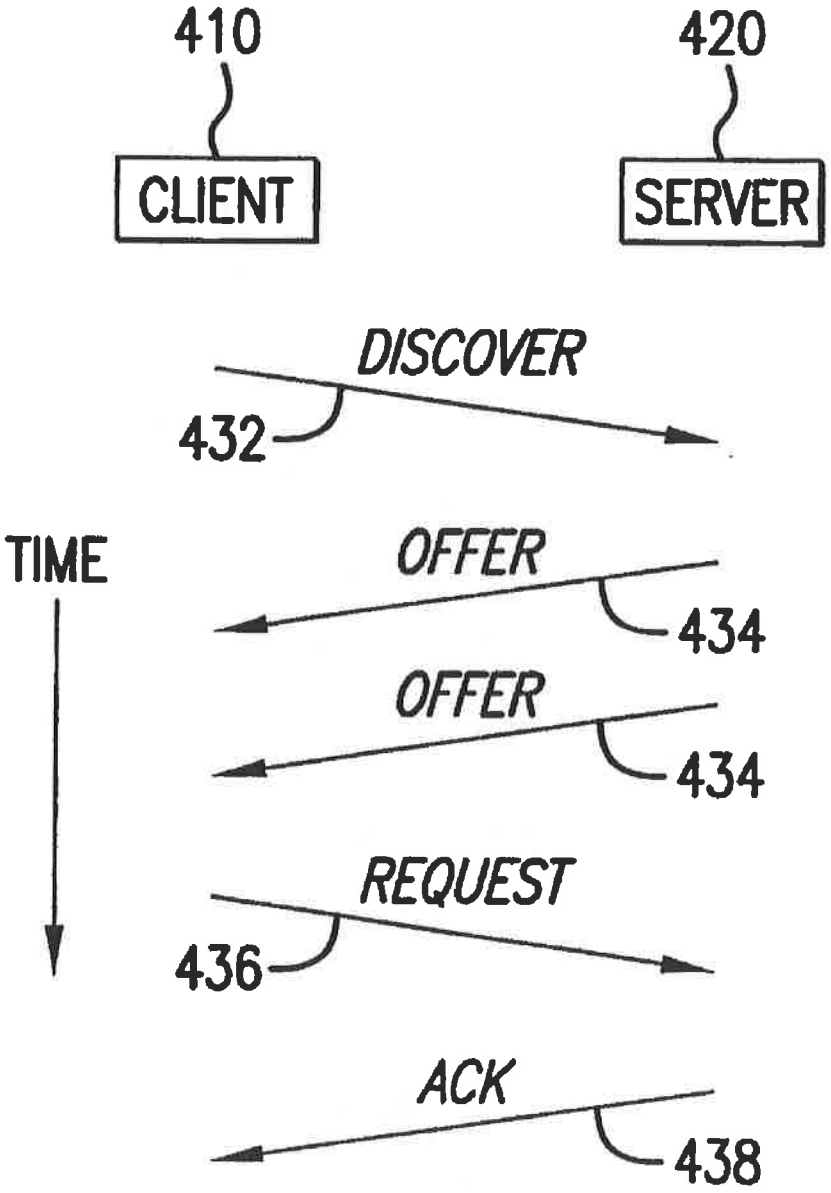


FIG. 4

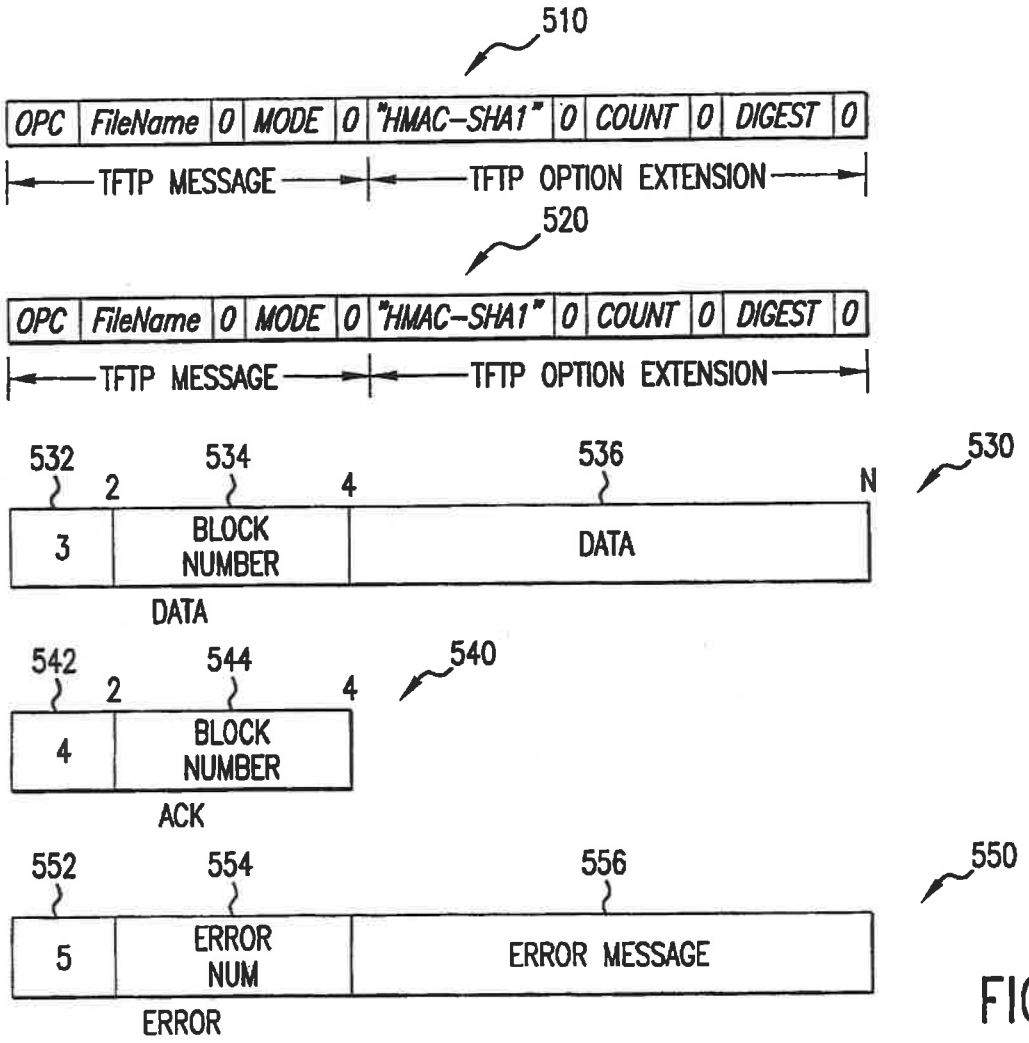


FIG.5

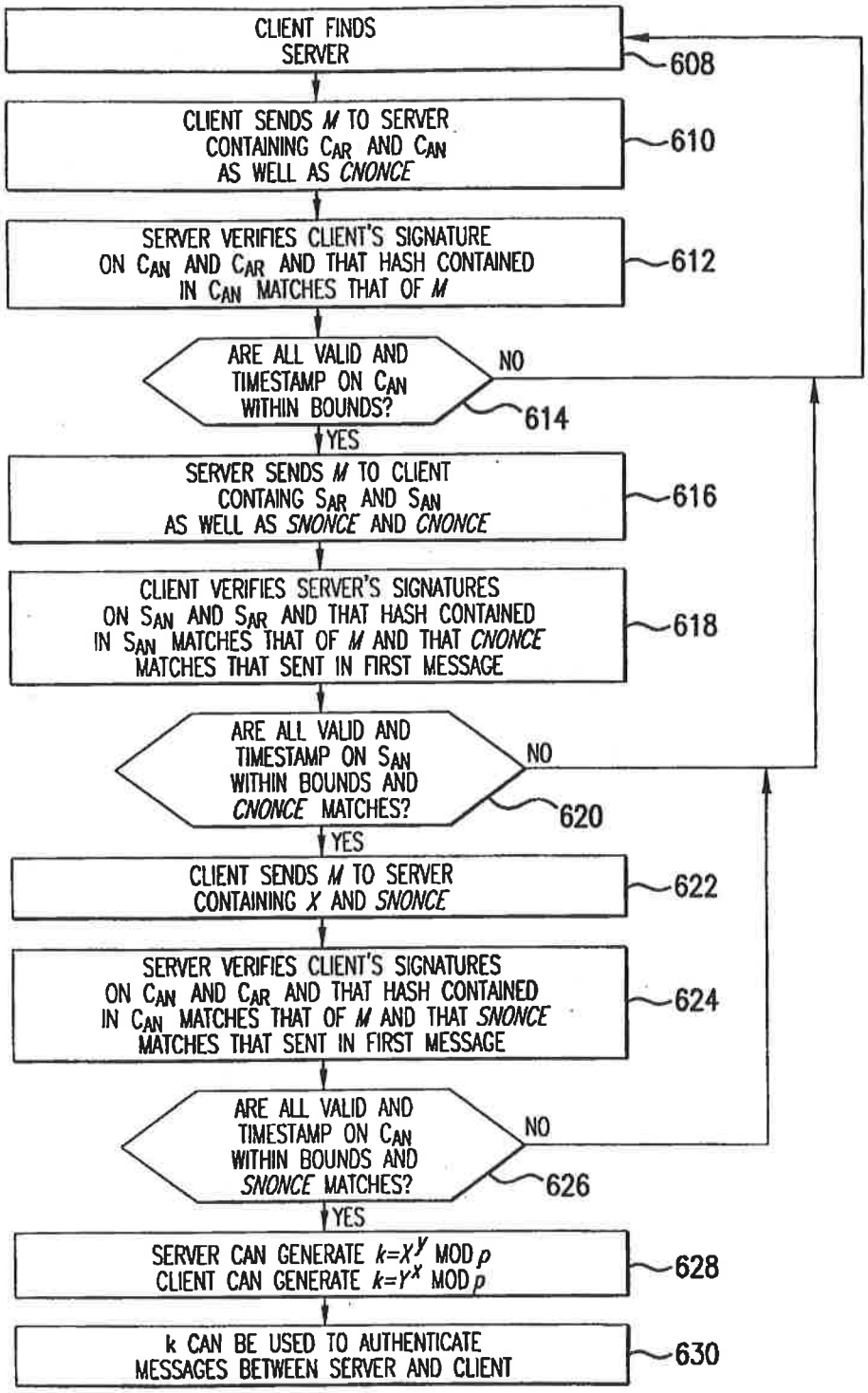


FIG.6

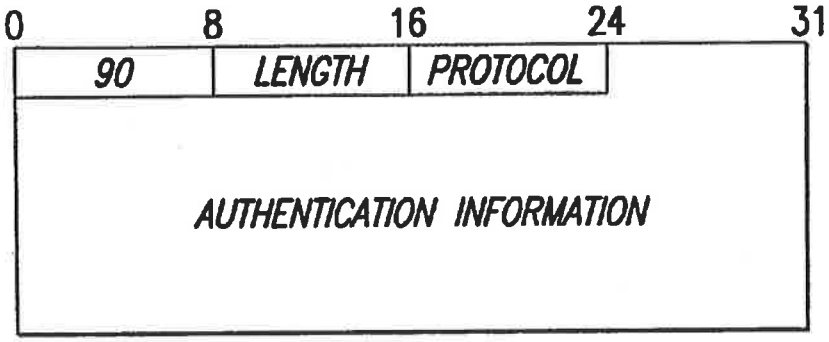


FIG.7

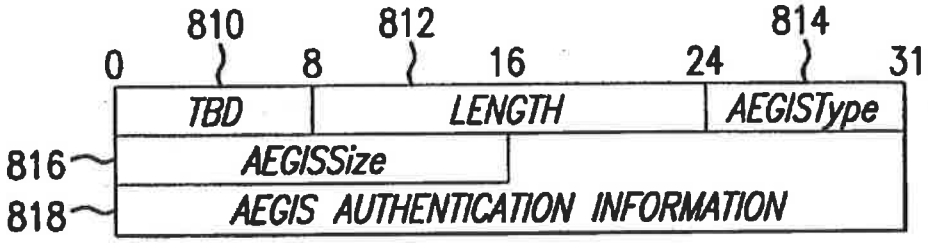


FIG.8

SECURE AND RELIABLE BOOTSTRAP ARCHITECTURE

CROSS-REFERENCE TO RELATED APPLICATION

This application claims the benefit of earlier filed U.S. provisional patent application Ser. No. 60/060,885 filed Oct. 2, 1997.

STATEMENT AS TO RIGHTS TO INVENTIONS MADE UNDER FEDERALLY-SPONSORED RESEARCH AND DEVELOPMENT

This invention was made with U.S. Government support under contracts #DABT63-95-C-0073, #N66001-96-C-852, and #MDA972-95-1-0013 awarded by the Advanced Research Project Agency. The U.S. Government has certain rights in the invention.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to an architecture for initializing a computer system and more particularly to a secure bootstrap process and automated recovery procedure.

2. Related Art

Systems are organized as layers to limit complexity. A common layering principle is the use of layers of abstraction to mark layer boundaries. A computer system is organized in a series of levels of abstraction, each of which defines a "virtual machine" upon which higher levels of abstraction are constructed. Each virtual machine presumes the correctness (integrity) of whatever virtual or real machines underlie its own operation. Under the presumption that the hardware comprising the machine (the lowest layer) is valid, integrity of a layer can be guaranteed if and only if: (1) the integrity of the lower layers is checked, and (2) transitions to higher layers occur only after integrity checks on them are complete. The resulting integrity "chain" inductively guarantees system integrity. When these suppositions are true, the system is said to possess integrity. When these conditions are not met, as they typically are not in the bootstrapping (initialization) of a computer system, no integrity guarantees can be made. Yet, these guarantees are increasingly important to diverse applications such as Internet commerce, security systems, and "active networks." However, it is surprising, given the great attention paid to operating system security today, that so little attention has been paid to the underpinnings required for secure operation, e.g., a secure bootstrapping phase for these operating systems. Without such a secure bootstrap the operating system kernel cannot be trusted since it is invoked by an untrusted process. Designers of trusted systems often avoid this problem by including the boot components (including but not limited to the ROM BIOS (Basic Input Output System), any expansion card ROMs, CMOS memory and NVRAM, the boot sector and the operating system kernel) in the trusted computing base (TCB). That is, the bootstrap steps are explicitly trusted. However, the present invention discloses that this provides a false sense of security to the users of the operating system, and more importantly, is unnecessary.

A number of attempts were made in the 1960s and 1970s to produce secure computing systems, using a secure operating system environment as a basis. However, an essential and unnecessary presumption of the security arguments for these designs was that system layers underpinning the operating system, whether hardware, firmware, or both, are

trusted. The first presentation of a secure boot process was done by Yee in *Dyad: A system for using physically secure coprocessors,* by J. Tygar and B. Yee, Technical Report CMU-CS-91 -140R, Carnegie Mellon University, May 1991. In Yee's model, a cryptographic coprocessor is the first to gain control of the system. Unfortunately, this is not possible without a complete architectural revision of most computer systems—even if the coprocessor is tightly coupled. Yee expanded his discussion of a secure boot in his thesis, see B. Yee, *Using Secure Coprocessors,* Ph.D. thesis, Carnegie Mellon University, 1994, but he continues to state that the secure coprocessor should control the boot process verifying each component prior to its use. Yee states that boot ROM modifications may be required, but since a prototype secure boot process was never implemented, more implementation questions are raised than answered by his discussion.

P.C. Clark presents, in *BITS: A Smartcard Protected Operating System,* Ph.D. thesis, George Washington University, 1994, a secure boot process for DOS that stores all of the operating system bootstrap code on a PCMCIA card. He does not address the verification of any firmware (system BIOS or expansion cards). Clark's model, however, does permit mutual cryptographic authentication between the user and the host which is an important capability. However, the use of the PCMCIA card containing all of the system boot files creates several configuration management problems, e.g., a system upgrade requires the reprogramming of all the cards in circulation, and since today many users have multiple operating systems on their personal computers a user needs a separate PCMCIA card for each operating system they wish to use.

B. Lampson, M Abadi, and M. Burrows also describe a secure boot model, in *Authentication in distributed systems: Theory and Practice,* ACM Transactions on Computer Systems, v10:265–310, November 1992, as an example for their authentication calculus. In the Lampson et al. model, the entire boot ROM is trusted, and they do not address the verification of expansion cards/ROMs. The Birlix Security Architecture, disclosed in *The Birlix security architecture,* by H. Härtig, O. Kowalski and W Kühnhauser, Journal of Computer Security, 2(1):5–21, 1993, proposes a model designed by Michael Gross that is similar to the Lampson et al. model. As a result, the Birlix model also suffers from the same problems. In both cases, the boot ROM is responsible for generating a public and private key pair for use in host based authentication once the operating system is running. The present invention, on the other hand, leaves any security related functions, beyond the boot process, to the operating system without loss of security. To do otherwise limits security choices for the operating system.

Two patents, U.S. Pat. No. 5,379,342 to Arnold ("the Arnold patent") and U.S. Pat. No. 5,421,006 to Jablon ("the Jablon patent") also present secure boot models. Both of these patents are similar in that the BIOS verifies the boot block before control is transferred and the boot block verifies the OS kernel before control is transferred. The Jablon patent continues to provide static integrity checks while the operating system is running, i.e validating the integrity of a program before execution. Another difference between the two patents is that the Arnold patent uses a Modification Detection Code, e.g. MD5, and Jablon uses public key cryptography. Both approaches, however, fail to verify the BIOS beyond the normal eight bit additive CRC, and both approaches also fail to verify expansion ROMs. The ROMs on add in boards are programs, and they are run during the boot process' of these two patents without

verification. Therefore, Jablon's and Arnold's approaches fail to provide a secure bootstrap process since neither approach verifies the BIOS and the ROMs.

Several anti-virus products also claim to create a secure boot process. A number of companies and BIOS vendors have anti-virus capabilities in their products. All concern themselves with the boot block only. Those products that run as an application over the operating system typically store a MDC for the boot block and check it when run. This detects changes to the boot block, but is susceptible to spoofing. The BIOS anti-virus protection simply alerts the user when a process is attempting to write to the boot block. The protection is ineffective when a protected mode operating system is running and a real mode application writes directly to the storage device. Finally, several vendors are now offering ROM based anti-virus protection. These products work by using an expansion ROM board that is executed during the boot process. The code on the ROM board checks the boot block against a previously stored MDC in order to detect changes. The vendors claim this prevents the possibility of spoofing the check as is possible when the check is done by an application. This is not entirely true, since the BIOS passes control to the ROM and if the BIOS has been reprogrammed to skip the ROM, control will never be passed to the ROM.

When a system detects an integrity failure, one of three possible courses of action can be taken. The first is to continue normally, but issue a warning. Unfortunately, this may result in the execution of either a corrupt or malicious component. The second is to not use or execute the component. This approach is typically called fail secure, and creates a potential denial of service attack. The final approach is to recover and correct the inconsistency from a trusted repository before the use or execution of the component. The first two approaches are unacceptable when the systems are important network elements such as switches, intrusion detection monitors, or associated with electronic commerce, since they either make the component unavailable for service, or its results untrustworthy.

None of the approaches mentioned above address a recovery process in the event of integrity failure or the secure recovery of bootstrap components. Previous efforts to provide recovery of bootstrap components have required human interaction, typically to insert a floppy disk containing the new component or to boot from a floppy disk. There are several reasons why this recovery method is inferior to the present invention. The first is that providing physical security for the floppy disk is extremely difficult. Users can take the disks wherever they like, and do whatever they like to them. The major shortcoming, however, in only using a boot disk is that none of the firmware is verified prior to use. Thus, a user can add or replace expansion boards into the system without any security controls, potentially introducing unauthorized expansion cards. Additionally, these efforts have only focused on repairing a single component of the entire process, i.e. only repairing the boot block, or the BIOS but not both. This is in contrast to the present invention which provides automatic recovery of all of the bootstrap components including ROM chips.

Finally, there have been several efforts at incorporating authentication into DHCP as is done in the AEGIS embodiment of the recovery process of the present invention. The first effort, disclosed in the expired RFC draft *Authentication for DHCP messages,* by R. Droms, November 1996, involves the use of a shared secret between the DHCP client and server. While this approach is secure, it severely limits the mobility of clients to those domains where a shared secret was previously established. Furthermore, the maintenance and protection of the shared secrets is a difficult process. Another effort at incorporating authentication into DHCP was by TIS. This proposal combines DHCP with DNSSEC, see D. Eastlake and C. Kaufman, *Dynamic Name Service and Security,* Internet RFC 2065, January 1997. This approach provides for the mobility of DHCP clients, but at a significant increase in cost in terms of complexity. The client implementation, in order to support this approach, must also include an implementation of DNSSEC. This will significantly increase the size of client code, possibly beyond the ROM size available to the client. Recently, Intel has proposed authentication support for DHCP, see Baiju V. Patel, *Securing DHCP,* Work in Progress, July 1997. Their proposal uses a two phase approach. In the first phase, the computer system boots normally using DHCP. The second phase begins after the system completes the DHCP process and uses ISAKMP to exchange a security association. This security association is then used to once again obtain the configuration information from the DHCP server using a secure channel, if such a channel can be established. This information is then compared to that obtained in the first phase. If they differ or a secure channel cannot be established, then the boot fails. The benefit of this approach is that it requires no changes to DHCP. The drawbacks are the same as the DNSSEC approach, discussed above, with the addition of two problems. The first is a possible race condition vulnerability during the time before the two configurations are compared. The second is that the approach does not protect against denial of service attacks.

SUMMARY OF THE INVENTION

The present invention discloses an architecture for initializing a computer system that ensures the integrity of the bootstrap process and provides reliability. Integrity is validated at each layer transition in the bootstrap process and a recovery process is included for integrity check failures. Ensuring the integrity is provided by the use of public key cryptography, a cryptographic hash function, and public key certificates. The present invention does this by constructing a chain of integrity checks, beginning at power-on and continuing until the final transfer of control from the bootstrap components to the operating system itself The integrity checks compare a computed cryptographic hash value with a stored digital signature associated with each component. Ensuring the integrity could also be done with the use of a modification detection code (MDC) with an increase in performance and a loss of security. Once an integrity failure is detected, the invention uses a secure protocol to inform a trusted repository that a failure has occurred and to obtain a valid replacement component. The secure protocol of the present invention can be based on well known networking protocols, such as DHCP (Dynamic Host Configuration Protocol) and TFTP (Trivial File Transfer Protocol), or on a custom protocol or various combinations of known protocols. Cryptographic algorithms are combined with the chosen protocols to add security to the recovery process, however if security is not a concern, then a less robust approach could be used.

The present invention can also be utilized to reduce the Total Cost of Ownership (TCO) of a personal computer, through automatically detecting and repairing integrity failures, thereby permitting the user to continue to work without the nuisance of a trouble call to support staff and the associated down time. A log can be created by the trusted repository of the present invention which can be monitored by a system administrator to identify workstations that

require "hands on" repairs, e.g. ROM failure, enabling the system administrator to schedule the work to be done when the user is not using the computer. The present invention also enables the bootstrap components to be automatically updated. One way this can be done is to limit the validity period of the cryptographic certificates associated with each component of the bootstrap process. When the certificate expires, the trusted repository of the present invention is contacted and either a new certificate is obtained, in the case where the component does not need an update, or a new component and certificate are obtained, in the case where a newer version of the component is available. This permits the system administrator to update all of the workstations from a central location without having to visit each individual computer system. A second approach is to add a hook to the BIOS to contact the trusted repository at the beginning of the bootstrap process of the present invention. The purpose of this contact is two fold. First, it permits a status monitoring of each workstation. Second, the contact allows centralized updates to be done in the following manner. When the server receives the "I am booting" message from the client, the server would check a database containing the configuration of the client. The Server would then compare that configuration with the current configuration desired for the client. If they are different, then the server would instruct the client to download the appropriate changes.

An alternate approach to provide a Secure and Reliable Bootstrap is to modify the above embodiments by moving the expansion ROM detection and verification routines into the operating system. The expansion ROMs can then be detected and verified by the operating system driver interface rather than the BIOS.

BRIEF DESCRIPTION OF THE FIGURES

FIG. **1***a* is a functional diagram of the functional layers of the typical IBM PC bootstrap process.

FIG. **1***b* is a flow chart showing the flow of the typical IBM PC bootstrap process.

FIG. **1***c* is structural diagram of a typical IBM PC architecture.

FIG. **2***a* is a functional diagram of the functional layers of the AEGIS embodiment of the bootstrap process of the current invention.

FIG. **2***b* is a flowchart showing the flow of the AEGIS embodiment of the bootstrap process of the current invention.

FIG. **2***c* is a flow chart showing the flow of an embodiment of the current invention in which the system configuration is automatically updated by the trusted repository at the beginning of the boot process of the present invention.

FIG. **2***d* is a flow chart showing the flow of an embodiment of the current invention in which the expansion card ROMs are detected and verified by the operating system driver interface rather than the BIOS.

FIG. **3** is a functional diagram of the format of a DHCP message.

FIG. **4** is a flow chart showing the flow of the initial DHCP message exchange between a client and a server.

FIG. **5** is a functional diagram of the format of the five TFTP messages.

FIG. **6** is a flow chart showing the flow of the message exchange between a client and a server to communicate and establish a shared secret of the present invention.

FIG. **7** is a functional diagram of the format of the DHCP Authentication Option Message.

FIG. **8** is a functional diagram of the modified DHCP Authentication Option Message of the current invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention is an architecture for initializing a computer system. AEGIS, an embodiment of the present invention is described below. AEGIS increases the security of the boot process by ensuring the integrity of bootstrap code. It does this by constructing a chain of integrity checks, beginning at power-on and continuing until the final transfer of control from the bootstrap components to the operating system itself. The integrity checks compare a computed cryptographic hash value with a stored digital signature associated with each component. This is accomplished through modifications and additions to the BIOS. The AEGIS architecture also includes a recovery mechanism for repairing integrity failures which protects against some classes of denial of service and modifications to components. In the AEGIS boot process, either the operating system kernel is started, or a recovery process is entered to repair any integrity failure detected. Once the repair is completed, the system is restarted to ensure that the system boots. This entire process occurs without user intervention.

In AEGIS, the boot process is guaranteed to end up in a secure state, even in the event of integrity failures outside of a minimal section of trusted code. A guaranteed secure boot process is defined in two parts. The first is that no code is executed unless it is either explicitly trusted or its integrity is verified prior to its use. The second is that when an integrity failure is detected the recovery process can recover a suitable verified replacement module. An added benefit of the recovery mechanism is the potential for reducing the Total Cost of Ownership (TCO) of a computer system by reducing trouble calls and down time associated with failures of the boot process.

From the start, AEGIS has been targeted for commercial operating systems on commodity hardware, making it a practical "real-world" system. To have a practical impact, AEGIS must be able to work with commodity hardware with minimal changes (ideally none) to the existing architecture. In the embodiment discussed below, the IBM PC architecture is selected as the platform because of its large user community and the availability of the source code for several operating systems. FIG. **1**(*c*) is a structural diagram of the typical IBM PC architecture. The computer system **1** includes one or more processors **4**. Processor **4** is connected to a expansion bus **6**. Computer system **1** also includes a main memory **8**, preferably random access memory (RAM) and a ROM BIOS **2**, which stores the system BIOS. Computer system may also include a secondary memory **10**. Secondary memory **10** may include, for example, a hard disk drive **12** and/or a removable storage drive **14**, representing a floppy disk drive, a magnetic tape drive, an optical disk drive, etc. The removable storage drive **14** reads from and/or writes to a removable storage unit **18** in a well known manner. Removable storage unit **18**, represents a floppy disk, magnetic tape, optical disk, etc. which is read by and written to by removable storage drive **14**. As will be appreciated, the removable storage unit **18** includes a computer usable storage medium having stored therein computer software and/or data.

Secondary memory **10** may also include other similar means for allowing computer programs or other instructions to be loaded into computer system **1**. Such means may include, for example, a removable storage unit **22** and an

interface **20**. Examples of such may include a removable memory chip (such as an EPROM or PROM) and associated socket, and other removable storage units **22** and interfaces **20** which allow software and data to be transferred from the removable storage unit **22** to computer system **1**.

Computer system **1** also includes a communications interface **24**. Communications interface **24** allows software and data to be transferred between computer system **1** and external devices. Examples of communications interface **24** may include a modem, a network interface (such as an Ethernet card), a communications port, a PCMCIA slot and card, etc. Software and data transferred via communications interface **24** are in the form of signals **28** which may be electronic, electromagnetic, optical or other signals capable of being received by communications interface **24**. These signals **28** are provided to communications interface **24** via a communications path (i.e., channel) **26**. This communication path **26** carries signals **28** and may be implemented using wire or cable, fiber optics, a phone line, a cellular phone link, an RF link and other communications paths.

The FreeBSD operating system is also used, but the AEGIS architecture is not limited to any specific operating system. Porting to a new operating system only requires a few minor changes to the boot block code so that the kernel can be verified prior to passing control to it. Since the verification code is contained in the BIOS, the changes will not substantially increase the size of the boot loader, or boot block.

I. Assumptions

The AEGIS model relies explicitly on three assumptions.

The first assumption is that the motherboard, processor, and a portion of the system ROM (BIOS) are not compromised, i.e. the adversary is unable or unwilling to replace the motherboard or BIOS. This assumption can be reduced by using a flash ROM such as the Intel 28F001BX-B which has an 8KB block that can be protected from reprogramming while the remainder of the ROM can be reprogrammed. Placing the bare essentials needed for integrity verification and recovery in this 8KB block provides a significant level of protection. The AEGIS model also depends on the integrity of a ROM expansion card which contains code for recovering components from a trusted network host. An alternative and less costly approach is to use the PROM available on most network cards in lieu of the additional expansion card.

The second assumption is the existence of a cryptographic certificate authority infrastructure to bind an identity with a public key, although no limits are placed on the type of infrastructure. An example of such an infrastructure is the infrastructure being established by Microsoft and Verisign for use with Authenticode.

The final assumption is that a trusted repository exists for recovery purposes. This repository may be a host on a network that is reachable through a secure communications protocol, or it may be a trusted ROM card located on the protected host.

II. AEGIS Boot Process

Every computer with the IBM PC architecture follows approximately the same boot process. This process can be divided into four functional layers, **110, 120, 130,140**, as is done in FIG. **1***a*, which correspond to phases of the bootstrap process, shown in FIG. **1***b*.

First layer **110** includes system BIOS **112** and corresponds to the first phase of the bootstrap process. The first phase of the boot process is the Power on Self Test or POST. POST is invoked, step **150**, in one of four ways:

1. Applying power to the computer automatically invokes POST causing the processor to jump to the entry point indicated by the processor reset vector.
2. Hardware reset also cause the processor to jump to the entry point indicated by the processor reset vector.
3. Warm boot (ctrl-alt-del under DOS) invokes POST without testing or initializing the upper 64K of system memory.
4. Software programs, if permitted by the operating system, can jump to the processor reset vector.

In each of the cases above, a sequence of tests are conducted, step **152**.

All of these tests, except for the initial processor self test, are under the control of system BIOS **112**.

Once system BIOS **112** has performed all of its power on tests, it begins searching a well known memory range for expansion card ROMs **122**, step **154**, which are identified in memory by a specific signature, such as the ROMs for the video card, the hard disk card and other devices. Once a valid signature is found by system BIOS **112**, step **156**, control is immediately passed to the corresponding expansion card ROM **122**. When the built in BIOS program on each expansion card ROM **122** completes its execution, step **158**, control is returned to system BIOS **112** and the search is continued for additional expansion card ROMs, step **154**.

The final step of the POST process calls the BIOS operating system bootstrap interrupt (Int 19h), step **160**. The bootstrap code first finds a bootable disk by searching the disk search order defined in the CMOS memory, step **162**. Once a bootable disk is found, step **164**, the bootstrap code loads primary boot sector **132** into memory, step **166**, and passes control to it, step **168**. The code contained in boot sector **132** proceeds to load operating system **142**, step **170**, or a secondary boot sector (not shown).

Ideally, the boot process would proceed in a series of levels with each level passing control to the next until the operating system kernel is running. Unfortunately, the IBM architecture uses a "star like" model, as shown in FIG. **1***a*, where control is passed to and from system BIOS **112** until finally passed on to boot sector **132**.

A. AEGIS BIOS Modifications

FIG. **2***a* shows the AEGIS BIOS modifications. In FIG. **2***a*, the boot process has again been divided up into several functional layers, **200, 210, 220, 230, 240, 250**, to simplify and organize the following discussion of the AEGIS BIOS modifications. Each increasing layer adds functionality to the system, providing correspondingly higher levels of abstraction. The lowest layer, first layer **200** contains the small section of trusted software, digital signatures, public key certificates, and recovery code AEGIS relies on throughout the boot process. The integrity of layer **200** is assumed to be valid. However, after initiating POST, step **260**, an initial checksum test is performed, step **262**, to identify PROM failures. Second layer **210** contains the remainder of the usual system BIOS code, and the CMOS memory. Third layer **220** contains all of the expansion cards, if any, and their associated expansion card ROMs **122**. Fourth layer **230** contains operating system boot sector(s) **132**. These are resident on the bootable device and are responsible for loading operating system kernel **142**. Fifth layer **240** contains operating system **142**, and sixth level **250** contains user level programs **252** and any network hosts **254**.

The transition between layers in a traditional boot process, as discussed above, is accomplished with a jump or call instruction without any attempt at verifying the integrity of the next layer. AEGIS, on the other hand, uses public key cryptography and cryptographic hashes to protect the transition from each lower layer to the next higher one, and its recovery process ensures the integrity of the next layer in the event of failures.

The pseudo code for the action taken at each layer, L, before transition to layer L+1, ie. at step **266, 274, 286,** and **292,** is:

```
if (IntegrityValid(L+1))) {
    GOTO (L+1);
} else {
    GOTO (Recovery);
}.
```

AEGIS modifies the boot process as shown in FIG. 1*a* by dividing system BIOS **112** into two logical sections. First section **202** contains the "trusted software", the bare essentials needed for integrity verification and recovery. Second, section **212** contains the remainder of the system BIOS **112** and the CMOS memory. First section **202** and second section **212** can be contained within a single flash ROM, such as the Intel 28F001BX-B which has an 8KB block that can be protected from reprogramming while the remainder of the ROM can be reprogrammed. Ideally, first section **202** is stored on this 8KB flash boot block to prevent tampering. Alternatively, an additional ROM card can be used to store the "trusted software", if memory constraints prevent the inclusion of the "trusted software" within the BIOS ROM. Similarly, if the computer system has a cryptographic coprocessor, such as the IBM 4758 PCI Cryptographic Coprocessor, or other preexisting cryptographic support, the cryptographic code and public key certificates could be removed from first section **202** and the coprocessor or other device could provide the cryptographic support for the integrity verification process.

First section **202** executes and performs the standard checksum calculation over its address space, step **262,** to protect against ROM failures. Following successful completion of the checksum, step **263,** the cryptographic hash of second section **212** is computed, step **264,** and verified against a stored signature, step **266.** If the signature is valid, control is passed to second section **212.**

Second section **212** proceeds normally with one change. Once second section **212** has performed all of its power on tests, it begins searching for expansion card ROMs **122,** step **268.** Once a valid signature is found by second section **212,** step **270,** control is passed to expansion card ROM **122.** However, prior to passing control to expansion ROM **122,** a cryptographic hash is computed, step **272,** and verified against a stored digital signature for the expansion card, step **274.** If the signature is valid, then control is passed to the expansion ROM **122** and it is executed. This process continues until the entire ROM space is searched. Once the verification of each expansion ROM **122** is complete, second section **212** passes control to the operating system bootstrap code, step **278.** The bootstrap code was previously verified as part of second section **212,** and thus no further verification is required. The bootstrap code finds a bootable device by following the CMOS search order, step **280,** and verifies boot sector **132,** step **286,** after computing a cryptographic hash of boot sector **132,** step **284.** Failure to find a bootable disk in step **280** may be resolvable through recovery process, step **298.**

If boot sector **132** is verified successfully, control is passed to it, step **288.** Finally, boot sector **132** computes a cryptographic hash of operating system kernel **142,** step **290,** and operating system kernel **142** is verified by boot sector **132,** step **292,** before passing control to it, step **294.** If a secondary boot sector is required (not shown), then it is verified by primary block sector **132** before passing control to it. Any integrity failures identified in the above process are recovered through a trusted repository, step **298,** as discussed below.

In the AEGIS boot process, either the operating system kernel is started, or a recovery process is entered to repair any integrity failure detected. Once the repair is completed, the system is restarted (warm boot) to ensure that the system boots. This entire process occurs without user intervention.

Ensuring the integrity could also be done with the use of a modification detection code (MDC) with an increase in performance and a loss of security.

B. Integrity Policy/Trusted Repository

The AEGIS integrity policy prevents the execution of a component if its integrity can not be validated. There are three reasons why the integrity of a component could become invalid. The integrity of the component could change because of some hardware or software malfunction, the integrity of the component could change because of some malicious act, or the component's certificate time stamp may no longer be valid. In each case, AEGIS attempts to recover from a trusted repository, step **298,** as discussed below. Should a trusted repository be unavailable after several attempts, then the client's further action depends on the security policy of the user. For instance, a user may choose to continue operation in a limited manner or may choose to halt operations altogether.

The AEGIS Integrity Policy can be represented by the following pseudo code:

```
StartOver:
if (ComponentCertificateValid) {
    if (ComponentIntegrityValid) {
        continue;
    } elseif (Recover (Component))
        goto StartOver;
    } else {
        User_Policy ();
    }
} else if (Recover(Certificate)) {
    goto StartOver;
    } else {
        UserPolicy ();
    }
}
```

The trusted repository can either be an expansion ROM board, not shown, that contains verified copies of the required software or it can be network host **254.**

The use of network host **254** as the trusted repository is accomplished through the addition of an inexpensive PROM board, and modifications to AEGIS ROM **256.** BIOS **112** and AEGIS ROM **256** contain the verification code, and public key certificates. AEGIS ROM **256** also contains code that allows the secure recovery of any integrity failures found during the initial bootstrap. In essence, the trusted software serves as the root of an authentication chain that extends to the operating system and potentially beyond to application software. If the component that fails its integrity check is a portion of BIOS **112,** then it must be recovered from AEGIS ROM **256.** The recovery process is a simple memory copy from the address space of AEGIS ROM **256** to the memory address of the failed component, in effect shadowing the failed component. A failure beyond BIOS **112** causes the system to boot into a recovery kernel contained on AEGIS ROM **256.** The recovery kernel contacts a "trusted" host through a secure protocol, as discussed below, to recover a verified copy of the failed component. The failed component is then shadowed or repaired, if possible, and the system is restarted.

Where network host **254** is the trusted repository, the detection of an integrity failure causes the system to boot into a recovery code contained on the AEGIS ROM **256**. The recovery code contacts a "trusted" host through the AEGIS recovery protocol, discussed below, to recover a signed copy of the failed component. The failed component is then shadowed or repaired, and the system is restarted (warm boot). Note that when the boot process enters the recovery procedure it becomes isomorphic to a secure network boot, except that in AEGIS only the needed bootstrap components are transferred. This fact is leveraged by adding authentication to the well known network protocols supporting the boot process (DHCP and TFTP) and using them as the recovery protocol, as discussed below.

In addition to ensuring that the system boots in a secure manner, AEGIS can also be used to maintain the hardware and software configuration of a machine. Since AEGIS maintains a copy of the signature for each expansion card (ideally the signature would be embedded in the firmware of the ROM), any additional expansion cards will fail the integrity test. Similarly, a new operating system cannot be started since the boot block and kernel would change, and the new boot block would fail the integrity test.

C. System Performance

In AEGIS, system integrity is preserved through the chain of integrity checks in the bootstrap process. The ideal authentication chain produced by each layer verifying the next can be represented by the recurrence:

$$I_o = \text{True},$$

$$I_{i+1} = \{I_i \Lambda V_i(L_{i+1}) \text{ for } 0 < i \leq 4$$

I_i is a boolean value representing the integrity of layer I, and Λ is the boolean and operation. V_i is the verification function associated with the i^{th} layer. V_i takes as its only argument the layer to verify, and it returns a boolean value as a result. The verification function performs a cryptographic hash of the layer, and compares the result to the value obtained from a stored signature for the layer. As stated earlier, the IBM PC does not lend itself to such a boot process. Instead, we alter the recurrence to:

$$I_o = \text{True},$$

$$I_{i+1} = \begin{cases} I_i \Lambda V_i(L_{i+1}) & \text{for } i = 0, 3, 4, \\ I_i \Lambda \sum_{i+1}^{n} V_i(L_{i+1}^i) & \text{for } i = 1, \\ I_i \Lambda V_{i-1}(L_{i+1}) & \text{for } i = 2. \end{cases}$$

Here, n represents the number of expansion boards in the system. Using the recurrence relation shown in this equation, the estimated increase in boot time (T_Δ), without integrity failures, between AEGIS and a standard IBM PC can be computed using the following equation:

$$T_\Delta = t(V_0(L_1)) + t\left(\sum_{i=1}^{n} V_1(L_2^i)\right) + t(V_1(L_3)) + t(V_3(L_4))$$

where t(op) returns the execution time of op. In estimating the time of the verification function, V_i, the BSAFE benchmarks for an Intel 90 Mhz Pentium computer, shown in the table below, are used.

Algorithm	Time
MD5	13,156,000 bytes/sec
RSA verify (512 bit)	0.0027 sec
RSA verify (1024 bit)	0.0086 sec
RSA verify (2048 bit)	0.031 sec

The cost of verification includes time required for computing a MD5 message digest, and the time required to verify the digest against a stored signature. Any signatures embedded in the public key certificate are ignored at the moment.

BIOS **112** is typically one megabit (128 kilobytes), and expansion ROMs **122** are usually 16 kilobytes, with some, such as video cards, as large as 64 kilobytes. For analysis purposes, it is assumed that one 64 kilobyte card and two 16 kilobyte cards are present. The size of boot sectors **132** for FreeBSD 2.2 (August 1996 Snapshot) are 512 bytes for the primary boot sector **132**, 6912 bytes for the secondary boot sector (not shown), and 1352 kilobytes for the size of GENERIC kernel **142**. Using the performance of MD5 from table 1, the time required to verify each layer using a 1024 but modulus is:

$t(V_0(L_1))$=0.0185 seconds

$t(V_1(L_2))$=0.0160 seconds

$t(V_1(L_3))$=0.018 seconds

$t(V_3(L_4))$=0.114 seconds

Summing these times gives T_Δ=0.1665 seconds which is insignificant compared to the length of time currently needed to bootstrap an IBM PC.

III. AEGIS Network Recovery Protocol

The AEGIS network recovery protocol combines protocols and algorithms from networking and cryptography to ensure the security of the protocol. The algorithms and protocols used are discussed first below and then the implementation of these algorithms and protocols in the AEGIS recovery process is discussed.

A. Digital Certificates

The usual purpose of a digital certificate with respect to public key cryptography is to bind a public key with an identity. While this binding is essential for strong authentication, it severely limits the potential of certificates, e.g., anonymous transactions. The most widely used certificate standard, the X.509 and its variants, provide only this binding. The X.509 standard, also, suffers from other serious problems in addition to its limited use. The most significant is ambiguity in the parsing of compliant certificates because of its use of the Basic Encoding Rules (BER). The encoding rules also require a great deal of space to implement, and the encoded certificates are usually large.

Because of the limits and problems with the X.509 certificate standard, a subset of the proposed SDSI/SPKI 2.0 certificate structure, see Carl M. Ellison, SDSI/SPKI BNF, *Private E-mail*, July 1997, can be used instead. The SDSI/SPKI format does not suffer from the same problems as X.509, and it offers additional functionality. The small subset of SDSI/SPKI needed for AEGIS is referred to here as SDSI/SPKI Lite. Below is the set of all possible strings of symbols that constitute legal programs in SDSI/SPKI Lite in extended Backus-Naur Form (BNF):

<byte-string>::<bytes>;

<bytes>::<decimal>":"{binary byte string of that length};

<cert>::"

("("cert"<issuer><subject><deleg>?<tag><valid>?")";

```
<client>::="("client"<cnonce>?<msg-hash>?")";
<cnonce>::="("cnonce"<byte-string>")";
<date>::<byte-string>;
<ddigit>::="0"|<nzdigit>;
<decimal>::<nzdigit><ddigit>;
<deleg>::="("propagate")";
<hash>::="("hash"sha1"<byte-string>")";
<issuer>::="("issuer"<issuer-name>")";
<issuer-name>::<principal>;
<msg-hash>::="("msg-hash"<hash>")";
<not-after>::="("not-after"<date>")";
<not-before>:: "("not-before"<date>")";
<nzdigit>::="1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9";
<obj-hash>::="("object-hash"<hash>")";
<principle>::<pub-key>|<hash-of-key>;
<pub-key>::="("public-key"<pub-sig-alg-id><s-expr>*
  <uri>?")";
<pub-sig-alg-id>::="dsa-sha1";
<s-expr>::="("<byte-string>")";
<server>::"("server"<dh-g>?<dh-p>?<dh-
  y>?<snonce>?<msg-hash>?")";
<signature>::="("signature"<hash><principle><byte-
  string>")";
<subject>::<principle>|<obj-hash>;
<tag>::="("tag")"|"("tag"<tag-body>")";
<tag-body>::<client>|<server>;
<valid>::<not-before>?<not-after>?;
```

SDSI/SPKI Lite provides for functionality beyond the simple binding of an identity with a public key. Identity based certificates require the existence of an Access Control List (ACL) which describes the access rights of an entity. Maintaining such lists in a distributed environment is a complex and difficult task. In contrast, SDSI/SPKI Lite provides for the notion of a capability. In a capability based model, the certificate itself carries the authorizations of the holder eliminating the need for an identity infrastructure and access control lists. In AEGIS, two capabilities, SERVER and CLIENT, are used with the obvious meanings. Additionally, AEGIS uses only three types of certificates. The first is an authorization certificate. An example of an AEGIS Authorization Certificate is shown below:

```
((cert (issuer (hash-of-key (hash sha1 caked)))
   (subject (hash-of-key (hash sha1 keyholderkey)))
   (tag (client))
   (not-before 03/29/97-0000)
   (not-after 03/29/98-0000))
  (signature (hash sha1 hashbytes)
   (hash-of-key (hash sha1 cakey))
   (sigbytes)))
```

This certificate, signed by a trusted third party or certificate authority, grants to the keyholder (the machine that holds the private key) the capability to generate the second type of certificate, an authentication certificate. The authentication certificate demonstrates that the client or server actually hold the private key corresponding to the public key identified in the authentication certificate. An example of an AEGIS client authentication certificate is shown below:

```
((cent (hash-of-key (hash sha1 clientkey)))
   (subject (hash-of-key (hash sha1 clientkey)))
   (tag (client (cnonce bytes) (msg-hash
     (hash sha1 bytes))))
   (not-before 09/01/97-0000)
   (not-after 09/01/97-0000)) (signature (hash sha1
     hashbytes)
   (public-key dsa-sha1 clientkey)
   (sigbytes)))
```


An example of an AEGIS server authentication certificate is shown below:

```
((cert (issuer (hash-of-key (hash sha1 serverkey)))
   (subject (hash-of-key (hash sha1 serverkey)))
   (tag (server (dh-g gbytes)
     (dh-p pbytes)
     (dh-Y ybytes)
     (msg-hash
     (hash sha1 hbytes))
     (cnonce cbytes)
     (snonce sbytes)))
   (not-before 09/01/97-0900)
   (not-after 09/01/97-0900)) (signature
   (hash sha1 hashbytes)
   (public-key dsa-sha1 serverkey)
   (sigbytes)))
```

In the above authentication certificate examples, the nonce field in the client authentication certificate is used along with a corresponding nonce in the server authentication certificate to ensure that the authentication protocol is "Fail Stop" detecting and to prevent active attacks such as a man-in-the-middle attack. The msg-hash field ensures that the entire message containing the certificates has not been modified. Using the msg-hash in the authentication certificate eliminates a signature and verification operation since the entire message no longer needs to be signed. The additional server fields are used to pass optional Diffie-Helman (DH) parameters, discussed below, to the client so that these parameters need not be global values. While clients are free to set the validity period of the authentication certificate to whatever they desire, it is expected that clients will keep the period short.

The third and final certificate format is the component signature certificate. An example of an AEGIS component certificate is shown below:

```
((cert (issuer (hash-of-key (hash sha1 approverkey)))
   (subject (hash sha1 hashtytes))
   (not-before 09/01/97-0000)
   (not-after 09/05/97-0000))
  (signature (hash sha1 hashbytes)
   (public-key dsa-sha1 approverkey)
  (sigbytes)))
```

This certificate is either embedded in a component or stored in a table. It is used with the AEGIS boot process described above.

Requiring each client to maintain a Certificate Revocation List (CRL) places a significant burden on the non-volatile storage of the client. Rather than use CRLs, the validity period of the certificates can be kept short, as in the SDSL/SPKI model, requiring the client to update the certificates when they expire. This serves two purposes beyond the ability to handle key revocation. First, the storage requirements for CRLs are eliminated. Second, the amount of system maintenance required of the client potentially can be reduced. Since the client must connect to the server on a regular basis to update the component certificates, the server can, at the same time, update the actual component as well if a new version is available.

B. Algorithms

1. Diffie-Rellman Key Agreement

The Diffie Hellman Key Agreement (DH), discussed in U.S. Pat. No. 4,200,770 to Hellman et al., incorporated herein by reference, permits two parties to establish a shared secret between them. Unfortunately, the algorithm as originally proposed is susceptible to a man-in-the-middle attack. The attack can be defeated, however, by combining DH with a public key algorithm such as DSA as proposed in the

Station to Station Protocol, discussed in *Authentication and Authenticated Key Exchanges*, W. Diffie, P. C. van Oorschot, and M. J. Wiener, *Designs, Codes and Cryptoraphy*, 2:107–125, 1992, incorporated herein by reference.

The DH algorithm is based on the difficulty of calculating discrete logarithms in a finite field. Each participant agrees to two primes, g and p, such that g is primitive mod n. These values do not need to be protected in order to ensure the strength of the system, and therefore can be public values. Each participant then generates a large random integer. Bob generates x as his large random integer and computes $X=g^x$ mod p. He then sends X to Alice. Alice generates g as her large random integer and computes $Y=g^y$ mod p. She then sends Y to Bob. Bob and Alice can now each compute a shared secret, k, by computing $k=Y^x$ mod p and $k=X^y$ mod p, respectively.

2. Digital Signature Standard

The Digital Signature Standard (DSS), discussed in *Digital Signature Standards*, Technical Report FIPS-186, U.S. Department of Commerce, May 1994, incorporated herein by reference, includes a digital signature algorithm (DSA) and a cryptographic hash algorithm (SHA1). DSA produces a 320 bit signature using the following parameters:

A prime, p, between 512 and 1024 bits in length. The size of the prime must also be a multiple of 64.

A 160 bit prime factor, q, of p−1.

g, where $g=h^{(p-1)/q}$ mod p and h is less than p−1 such that g is greater than 1.

x, where x is less than q.

y, where $y=g^x$ mod p.

The parameters p, q, and g are public. The private key is x, and the public key is y. A signature of a message, M, is computed in the following manner. The signer generates a random number, k, that is less than q. They then compute $r=(g^k \text{mod} p)$ mod q, and $s=(k^{-1}(SHA1(M)+xr))$ mod q. The values r and s, each 160 bits in length, comprise the signature. The receiver verifies the signature by computing:

$$w=s^{-1} \text{ mod } q$$

$$u_1=(SHA1(M)*w) \text{ mod } q$$

$$u_2=(r*w) \text{ mod } q$$

$$v=((g^{u1}*y^{u2}) \text{ mod } p) \text{ mod } q.$$

The signature is verified by comparing v and r. If they are equal, then the signature is valid.

3. SHA1 Message Authentication Code

Message Authentication Codes (MAC) utilize a secret, k shared between the communicating parties and a message digest. AEGIS uses the Secure Hash Algorithm (SHA1), discussed in *Secure Hash Standard*, Technical Report FIPS-180-1, U.S. Department of Commerce, April 1995 (also known as 59 Fed Reg 35317 (1994)) incorporated herein by reference, and the HMAC, described in *HMAC: Keyed-Hashing for Message Authentication*, Internet RFC 2104, February 1997, incorporated herein by reference. The MAC is defined as:

SHA1 (*k*XOR opad, SHA1 (*k*XOR ipad, *M*)),

where M is the message or datagram, opad is an array of 64 bytes each with the value 0x5c, and ipad is an array of 64 bytes each with the value 0x36. k is zero padded to sixty four bytes. The result of this MAC is the 160-bit SHA1 digest.

C. Protocols

1. DHCP—Dynamic Host Configuration Protocol

The DHCP protocol, discussed in *Dynamic Host Configuration Protocol*, Internet RFC 2131, March 1997, incorporated herein by reference, provides clients the ability to configure their networking and host specific parameters dynamically during the boot process. The typical parameters are the IP addresses of the client, gateways, and DNS server. DHCP, however, supports up to 255 configuration parameters, or options. Currently approximately one hundred options are defined for DHCP, see *DHCP Options and BOOTP Vendor Extensions*, Internet RFC 2132, March 1997, incorporated herein by reference. One of these options is an authentication option which is described below.

The format of a DHCP message is shown in FIG. **3**. The first field in the DHCP message is the opcode **310**. Opcode **310** can have one of two values, 1 for a BOOTREQUEST message, and 2 for a BOOTREPLY message. The next field, htype **312**, is the hardware address type defined by the "Assigned Numbers" RFC, see J. Reynolds and J. Postel, *Assigned Numbers*, Interent RFC 1700, October 1994, incorporated herein by reference. The field hlen **314** indicates the length of the hardware address. The field hops **316** is set to zero by the client and used by BOOTP relay agents to determine if they should forward the message. The field xid **318** is a random number chosen by the client. Its use is to permit the client and the server to associate messages between each other. The field secs **320** is set by the client to the number of seconds elapsed since the start address acquisition process. Currently, only the leftmost bit of the flags **322** field is used to help solve an IP multicast problem. The remaining bits must be zero. The field ciaddr **324** is the client address if the client knows it already. The field yiaddr **326** is "your" address set by the server if the client did not know its address or had a bad one. The field giaiddr **330** is the relay agent address., chaddr **332** is the client's hardware address, same **334** is an optional null terminated string containing the server's name, and file **336** is the name of the boot file. In AEGIS, this is the name of the component to recover. Finally, options **338** is a variable length field containing any options associated with the message.

The initial message exchange between client **410** and server **420** is shown in FIG. **4**. Client **410** begins the process by sending a DHCPDISCOVER message as a broadcast message on its local area network, step **432**. The broadcast message may or may not be forwarded beyond the LAN depending on the existence of relay agents at the gateways. Any or all DHCP servers **420** respond with a DHCPOFFER message, step **434**. Client **410** selects one of the DHCPOFFER messages and responds to that server **420** with a DHCPREQUEST message, step **436**. Server **420** acknowledges the DHPREQUEST message with a DHCPACK, step **438**.

In addition to providing networking and host specific parameters, DHCP can provide the name and server location of a bootstrap program to support diskless clients. After the client receives the IP address of the boot server and the name of the bootstrap program, the client uses TFTP, discussed below, to contact the server and transfer the file.

2. TFTP-Trivial File Transfer Protocol

TFTP, discussed in *The TFTP Protocol (revision 2)*, by K. R. Sollins, Interent RFC 1350, July 1992, incorporated herein by reference, was designed to be simple and small enough to fit in a ROM on a diskless client. Because of this, TFTP uses UDP, User Datagram Protocol, rather than TCP, Transport Control Protocol, with no authentication included in the protocol. TFTP does, however, have an option capa-

bility similar to DHCP, see G. Malkin and A. Harkin, *TFTP Option Extension,* Internet RFC 1782, March 1995, incorporated herein by reference.

TFTP has five unique messages that are identified by a two byte opcode value at the beginning of the packet. Read Request (RRQ) **510** and Write Request (WRQ) **520** packets, opcodes **1** and **2** respectively, share the same format, as shown in FIG. **5**. Data (DATA) packet **530**, as shown in FIG. **5**, contains three fields. The first field, **532**, is the two byte opcode, **3** for DATA. Following the opcode is a two byte field, **534**, containing the block number of the data, beginning at 1 and increasing. The third and final field of the packet, **536**, contains the actual block of data transferred. Typically, the block size is 512 bytes. However, the size can be increased through the use of the TFTP options. Where the block is smaller than the blocksize, this identifies the packet as the final DATA packet **530**. Each DATA packet **530** is acknowledged by a four byte ACK packet **540**, opcode **4**, as shown in FIG. **5**, containing the opcode, **542**, and the acknowledged block number, **544**. The final packet, opcode **5**, is ERROR packet **550** with three fields, as shown in FIG. **5**. The first, **552**, is the two byte opcode. The second, **554**, is a two byte error code, and the final field, **556**, is a zero terminated netascii string containing an error message.

A TFTP session for reading/downloading a file begins with the client **410** sending an RRQ packet **510** to the sever **420** and receiving either a first DATA packet **530** in response, or an ERROR packet **550** if the request was denied. The client **410** responds with an ACK packet **540**, and the process continues until the file is transferred.

D. Implementation

Client **410** (AEGIS) and Server **420** (Trusted Repository) wish to communicate and establish a shared secret after authenticating the identity of each other. There has been no prior contact between Client **410** and Server **420** other than to agree on a trusted third party (CA), or a public key infrastructure, to sign their authorization certificates, C_{AR} and S_{AR}. Server **420** and Client **410** also need to have a copy of the trusted third party's public key, P_{CA}, for use in verifying each other's authorization certificates. The process Client **410** and Server **420** follow is shown in FIG. **6**. First, Client **410** sends a message out over the network to locate a Server **420**, step **608**. When Server **420** is found, Client **410** sends a message, M, to Server **420** containing the Client's authorization certificate, C_{AR}, authentication certificate, C_{AN}, and nonce, cnonce, step **610**. Server **420** receives the message, M, and verifies Client's signature on the authentication certificate, C_{AN} and that the hash contained in the authentication certificate, C_{AN}, matches that of the message, M, step **612**. The signature of the CA on the authorization certificate, C_{AR}, is also verified. If all are valid and the timestamp on the authentication certificate, C_{AN}, is within bounds, step **614**, then Server **420** sends to Client **410** a message, M, containing its authorization certificate, S_{AR}, and authentication certificate, S_{AN}, step **616**. Server's authentication certificate, S_{AN}, may include the optional DH parameters, g and p, and Y, where $Y=g^y \bmod p$. If the DH parameters are not included in the certificate, then default values for g and p are used. Server's nonce, snonce, and Client's nonce, cnonce, are also included in message, M Client **410** receives message, M, and verifies the signatures on the authentication certificate, S_{AN}, and authorization certificate, S_{AR}, and that the hash in Server's authentication certificate matches the message hash, and that cnonce matches that sent in the first message, step **618**. If all are valid and the timestamp value of the authentication certificate is within bounds and cnonce matches that sent in the first message, step **620**, then Client **410** sends a signed message to Server **420** containing its DH parameter X where $Y=g^x \bmod p$ and Server's nonce snonce, step **622**. Server **420** receives the message and verifies the signatures and that snonce matches that sent in its previous message, step **624**. If both are valid, step **626**, then Server **420** can generate the shared secret, k, using $k=X^y \bmod p$ and Client **410** similarly can generate the shared secret, k, using $k=Y^x \bmod p$, step **628**. The shared secret, k, can now be used to authenticate messages between Server **420** and Client **410**, step **630**, until such time as both agree to change k. The use of the authentication certificates, C_{AN} and S_{AN}, assists in ensuring that the protocol is "Fail Stop" through the use of nonces and a short validity period for the certificate. The use of snonce also permits Server **420** to reuse Y over a limited period. This reduces the computational overhead on Server **420** during high activity periods. The potential for a TCPSYN like denial of service attack, is mitigated in the same manner by the authentication certificates. The authorization certificates, C_{AR} and S_{AR}, also prevent Clients **410** from masquerading as Server **420** because of the client/server capability tag. This is a benefit not possible with X.509 based certificates.

Subsequent messages, e.g. TFTP messages, use the SHA1 HMAC defined above with a one up counter to prevent replays. The counter is initially set to zero when the shared secret, k, is derived.

Where validation steps, **614**, **620**, and **626** are unsuccessful in the above process, the process starts over with Client **410** searching for a Server **420**, step **608**.

E. Using DHCP/TFTP as the AEGIS Recovery Protocol

1. DHCP Authentication Option

DHCP is extensible through the use of the variable length options field **338** at the end of each DHCP message. The format of the message is shown in FIG. **7**. The DHCP authentication option was designed to support a wide variety of authentication schemes by using the single byte protocol and length fields. Unfortunately, a single byte value for the size in octets of authentication information is too small for the AEGIS authentication information. To solve this problem, the choices were to either violate the current DHCP options standard and use a two byte size field and potentially cause interoperability problems, or to place an additional restriction on the AEGIS authentication packet, requiring it to be the last option on any DHCP packet. The latter has been selected in this embodiment. Using this and a unique AEGIS option number permits interoperability with current DHCP servers.

Since the authentication option message format shown in FIG. **7** is not used, a new DHCP option format for AEGIS Authentication must be defined, as shown FIG. **8**. The AEGIS option uses the same basic format as the normal DHCP format., the only difference is the use of a two byte size field. Embedded in the data portion, **818**, of the option are the AEGIS certificates, and other data as required. These fields are identified through the use of a one byte AEGIS type, **814**, followed by a two byte size field, **816**. The AEGIS Authentication format is shown in FIG. **8**. The different AEGIS types are shown in the table below:

Type	Value
Authorization Certificate	0
Client Authorization Certificate	1
Server Authorization Certificate	2

-continued

Type	Value
Component Authentication Certificate	3
X value	4
snonce	5
signature	6
SHA1 MAC	7

2. Adding Authentication to TFTP

A new TFTP option is also defined, HMAC-SHA1, that uses the HMAC defined above along with a 32 bit one up counter for use with the TFTP Read (RRQ) **510** and Write (WRQ) **520** requests. The format of a RRQ **510** or WRQ **520** packet with the HMAC option is shown in FIG. **5**. The counter is two bytes in length, and its purpose is to prevent replay attacks. Both Client **410** and Server **420** initialize the count to zero immediately after k is derived from the protocol shown in FIG. **6**.

The TFTP option extension, however, is not defined for TFTP DATA **530** or ERROR **540** packets. Therefore, those packets must be extended in the same manner as was done with RRQ packet **510** and WRQ packet **520** above. The extended TFTP packet formats are shown in FIG. **5**.

Another TFTP implementation problem is how to handle the "lock-step" nature of the protocol and still prevent replays. The solution adopted here is to provide a narrow window for an adversary to obtain a copy of the file from Server **420** without proper authentication by replaying the message to Server **420** before Client's next message. The benefits of this approach, not having to change the TFTP protocol other than a small message format change, outweigh the potential problems associated with dramatically changing the protocol.

3. The AEGIS Recovery Protocol

Once authentication is added to DHCP and TFTP, AEGIS can use them without further modifications as its recovery protocol. In AEGIS, the client follows the DHCP protocol, as shown in FIG. **4**, but adds to the DHCPDISCOVER message, step **432**, the name of the required component needed followed by the SHAI hash of the component in the boot file name field, **336**. Once the DHCP protocol is completed and the shared secret established, the AEGIS client contacts the trusted repository using TFTP with authentication and downloads the new component.

Performance estimates can be made using the times shown in the table below, for results generated using a 200 Mhz PentiumPro with 32 MB of memory.

Algorithm	Time
SHA1	6.1 MB/sec
DSA Verify (1024 bit)	36 msec
DSA Sign (1024 bit)	23 msec
Generate X,Y (1024 bit)	22 msec
Generate k (1024 bit)	71 msec

For the purposes of these estimates, it is assumed that each DHCP message is three kilobytes in length. The cost of hashing the first and second message for comparison to the hash contained in the authentication certificate is negligible and therefore not included in the estimates below.

The initial authentication exchange includes the first three DHCP messages, DHCPDISCOVER, step **432**, DHCPOFFER, step **434**, and DHCPREQUEST, step **436**. DHCPDISCOVER, step **432**, requires Client **410** to perform one signature operation, and Server **420** must perform two verify operations. Thus, the total cost of this message is 95 msec. The DHCPOFFER message, step **434**, requires Server **420** to generate Y and perform one signature operation. Client **410** must perform two verify operations. This results in a message cost of 117 msec. The final message, DHCPREQUEST, step **436**, requires Client **410** to generate X and k, and perform one signature operation. Server **420** must perform one verify operation, and generate k resulting in a message cost of 107 msec. Summing the cost of these three messages gives a total cost of 319 msec.

While the above time may seem too high a cost to pay for security, the total time is small when compared to the total time spent booting a computer system. It is unlikely that users will see the increase in time required to perform the authentication.

Subsequent messages use the MAC described earlier, and will likely (in a LAN situation) be bounded by the speed of SHA1, 6.1 MB/sec.

IV. Conclusion

The approach of this invention is to ensure the integrity of the bootstrap process and provide reliability. Ensuring the integrity is provided in the above embodiment by the use of public key cryptography, a cryptographic hash function, and public key certificates. Ensuring the integrity could also be done with the use of a modification detection code (MDC) with an increase in performance and a loss of security. Once an integrity failure is detected, the invention uses a secure protocol to inform a trusted repository that a failure has occurred and to obtain a valid replacement component. While the above embodiment uses well known networking protocols, a custom protocol or various combinations of known protocols could also be used. Additionally while the above embodiment employs cryptographic algorithms for security purposes, if security is not a concern a less robust approach could be used.

The recovery process discussed above is also easily generalized to applications other than the boot process of the present invention, such as standardized desktop management and secure automated recovery of network elements such as routers or "Active Network" elements.

While AEGIS will serve as a strong foundation for future security measures, it also has the potential for reducing the Total Cost of Ownership (TCO) of IBM personal computers. Automatically detecting and repairing integrity failures permits the user to continue to work without the nuisance of a trouble call to the support staff and the associated down time spent waiting. A system administrator can monitor the log of the AEGIS trusted repository and identify those workstations that require "hands on" repairs, e.g. ROM failure, and schedule the work to be done when the user is not using the computer. This permits the system administrator to schedule a trouble call rather than react to it. This greatly reduces the stress on the users and the administrators. AEGIS can also offer the ability to provide automatic updates of the bootstrap components. There are two possible approaches. The first limits the validity period of the cryptographic certificates associated with each component of the bootstrap process. When the certificate expires, AEGIS contacts the trusted repository and either obtains a new certificate in the case where the component does not need an update, or a new component and certificate in the case where a newer version of the component is available. This permits the system administrator to update all of their workstations from a central location without having to visit each individual computer system. The second approach adds a hook, step **261** in FIG. 2*c*, to the BIOS to contact the trusted repository

after initiating POST, step **260**. As is shown in FIG. **2***c*, the remainder of the boot process follows the same procedure as that described for FIG. **2***b* above. The purpose of this contact is two fold. First, it permits a status monitoring of each workstation. Second, the contact allows centralized updates since the trusted repository can respond back with a "I need to update you" message. Step **261** would be done in the following manner. When Server **420** receives the "I am booting" message from Client **410**, Server **420** would check a database containing the configuration of Client **410**. Server **420** would then compare that configuration with the current configuration desired for Client **410**. If they are different, then Server **420** would instruct Client **410** to download the appropriate changes. AEGIS can also "lock down" the configuration of a system and prevent users from changing operating systems, adding PROM cards, or modifying BIOS settings. Greatly reducing the problems caused by well intentioned, but technically unsophisticated, users.

An alternate approach to provide a Secure and Reliable Bootstrap is to move the expansion ROM detection and verification routines, steps **268**,**270**,**272**, and **274** in FIG. **2***b*, into operating system **142**. As is shown in FIG. **2***d*, after boot sector **132** loads operating system **142**, step **294**, expansion ROMs **122** can be searched for, step **295**, detected, step **296**, and verified, step **298**, using a computed cryptographic hash, step **297**, by the operating system driver interface rather than the BIOS. The initialization of operating system **142** is then allowed to continue, step **299**.

While a number of embodiments of the present invention have been described above, they should be understood to have been presented by way of example, not limitation. It will be apparent to persons skilled in the art that various changes in form and detail can be made therein without departing from the spirit and scope of the invention. Thus the present invention should not be limited by the above described exemplary embodiments, but should be defined in accordance with the following claims and their equivalents.

What is claimed is:

1. An architecture for initializing a computer system comprising:

a processor;

an expansion bus coupled to said processor;

a memory coupled to said expansion bus, said memory storing a system BIOS for execution by said processor upon power up of the computer system;

a plurality of boot components coupled to said expansion bus and accessed by said processor when said system BIOS is executed;

a trusted repository coupled to said expansion bus; and

means for verifying the integrity of said boot components and said system BIOS wherein integrity failures are recovered through said trusted repository.

2. An architecture for initializing a computer system according to claim **1**, wherein said trusted repository is an expansion ROM coupled to said expansion bus.

3. An architecture for initializing a computer system according to claim **1**, wherein said trusted repository is a host computer communicating with said computer system through a communications interface coupled to said expansion bus.

4. A method for initializing a computer system comprising the steps of:

(1) invoking a Power on Self Test (POST);

(2) verifying the integrity of a system BIOS;

(3) verifving the integrity of a boot component; and

(4) when said boot component fails, recovering said failed boot.

5. A method according to claim **4**, wherein step (1) further comprises the step of performing a checksum calculation over the address space of a trusted memory location.

6. A method according to claim **4**, wherein step (3) comprises the steps of:

(a) computing a cryptographic hash value for said boot component; and

(b) comparing said cryptographic hash value with a digital signature associated with said boot component stored in a trusted memory location.

7. The method of claim **4**, wherein step (4) employs a secure protocol to obtain a replacement boot component from a trusted repository to replace said failed boot component.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 6,185,678 B1
APPLICATION NO.	: 09/165316
DATED	: February 6, 2001
INVENTOR(S)	: William A. Arbaugh et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 22, line 24 replace "verifving" with -- verifying --

Column 22, line 26 insert -- component – after "failed boot"

Signed and Sealed this
Sixth Day of November, 2012

David J. Kappos
Director of the United States Patent and Trademark Office